FORENSICS V2 LAB SERIES

Lab 05: File System

Document Version: 2021-01-14

Contents

Introduction

This module will help the student understand what file systems are. They will also learn the limitations and advantages of each one and how to identify the differences.

Objectives

- Understand what the most popular file systems are
- Learn how to identify each file system using hex editors
- Learn what volume serial numbers are and how to decode them
- Learn to decode volume creation dates and times

Lab Topology

Lab Settings

The information in the table below will be needed to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	IP Address / Subnet Mask	Account (if needed)	Password (if needed)
Caine	172.16.16.30	caine	Train1ng$
CSI-Linux	172.16.16.40	csi	csi
DEFT	172.16.16.20	deft	Train1ng$
WinOS	172.16.16.10	Administrator	Train1ng$

1 Getting to Know HxD Hex Editor

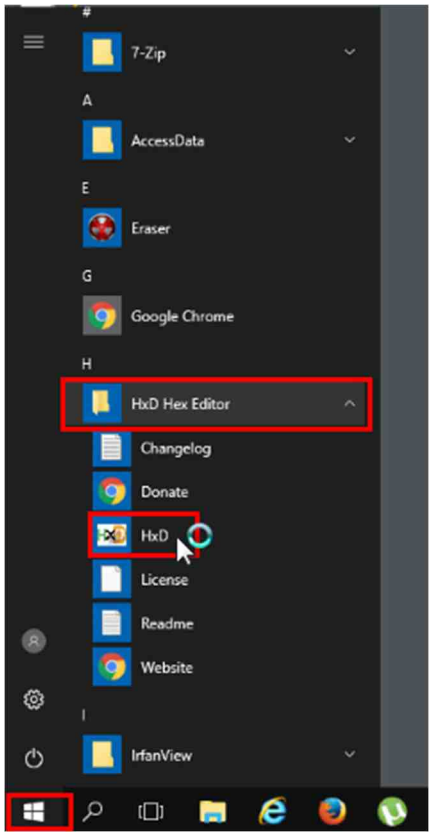
The ability to understand partitions and file systems is extremely important in performing digital examinations. Partitions determine how much data you can access, and file systems determine how that data is handled. The two are closely related but very different. In this lab, we will use a very useful FreeWare tool called HxD Hex Editor and Disk Editor[1] to access and parse data that will help us identify and understand partitions and file systems better.

1. To begin, launch the WinOS virtual machine to access the graphical login screen.
   a. Select Send CTRL+ALT+DEL from the dropdown menu to be prompted with the login screen.

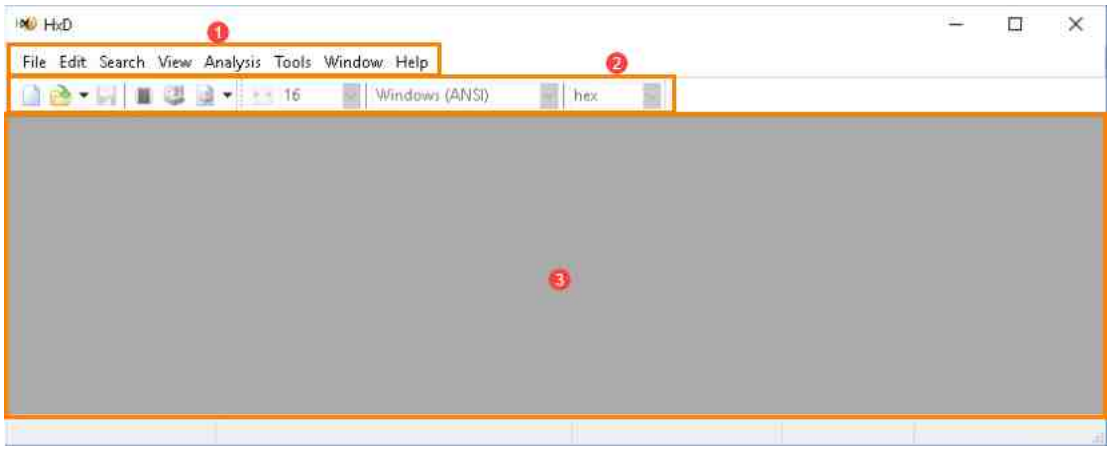
   b. Log in as `Administrator` using the password: `Train1ng$`.

[1] https://mh-nexus.de/en/hxd/

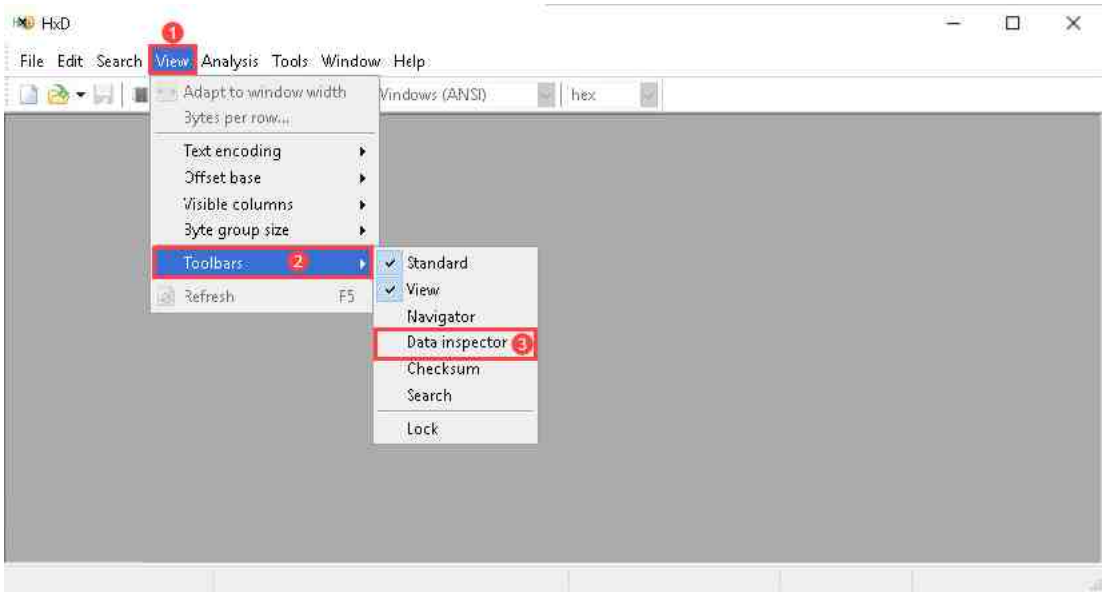
2. Once you are logged into the VM, launch the HxD Hex Editor program from the Windows Start menu by navigating to Start Menu > HxD Hex Editor. Alternatively, you can open HxD Hex Editor from the Desktop by double-clicking the icon called HxD:



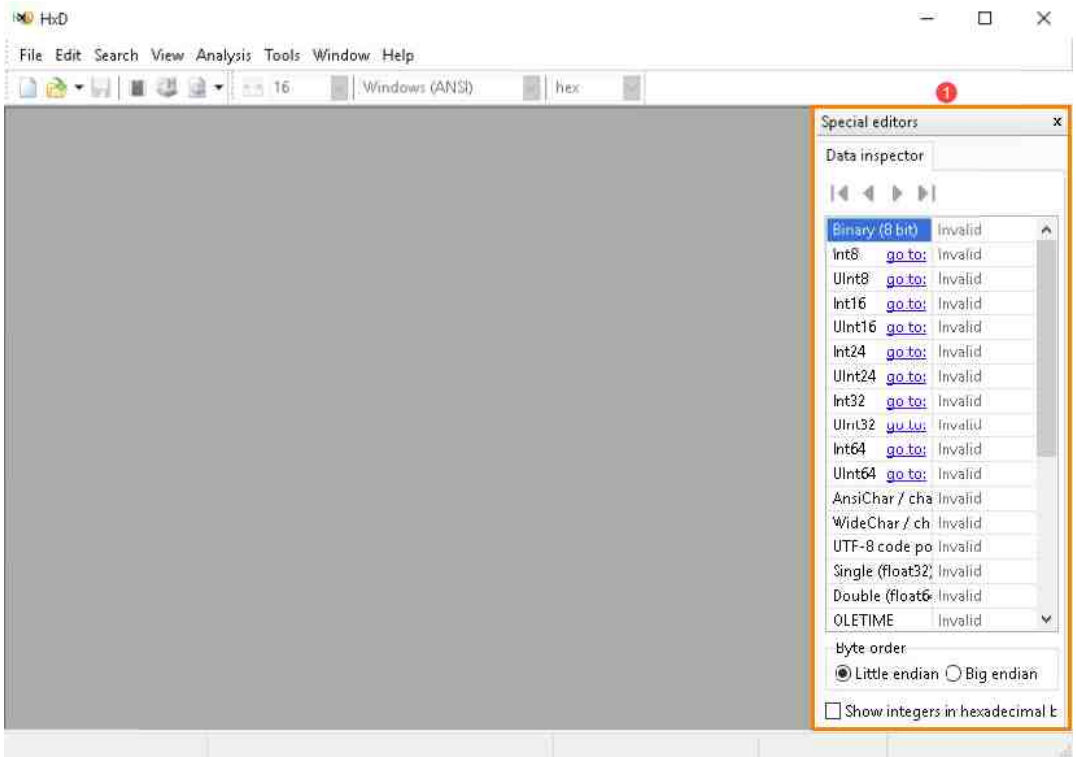
3. Once you have HxD opened, you will see the main interface, as seen in the screenshot below. Let us look at the GUI first. The main GUI window has 2 toolbars by default.
   a. They are highlighted as items 1 and 2. The toolbar highlighted as item 2 contains quick use icons that can also be found in the Menu bar highlighted as item 1.
   b. The pane highlighted as item 3 is the area where the files' contents are displayed in hexadecimal and text view.



c. There is a feature called the Data Inspector tab found in the Special editors pane that we will need for this exercise. Let us check if it is open by navigating to View > Toolbars as seen in items 1 and 2. Once there, review the submenu that appears. If the Data Inspector option seen in item 3 does not have a checkmark beside it, then click it. If it does have a checkmark, then exit the menu by clicking on an empty area outside of the menu.

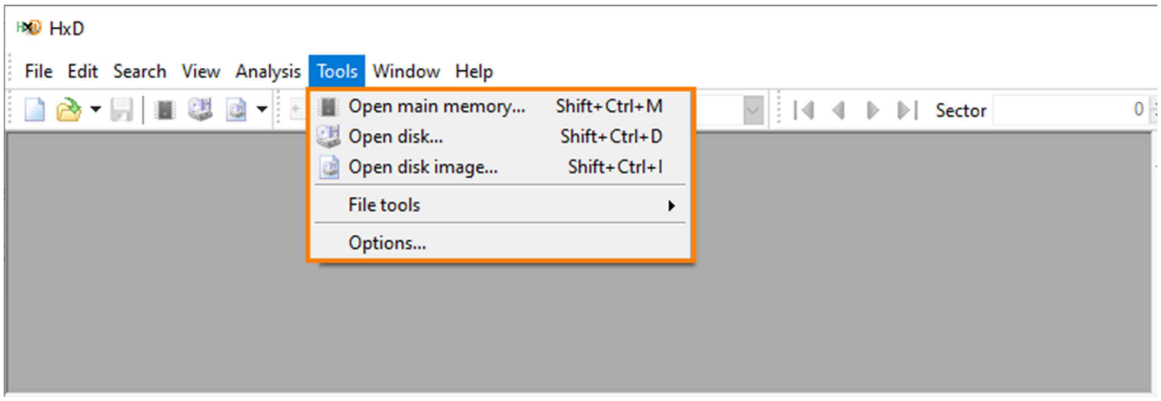


d. Now the Special editors pane will appear, and you will see the Data inspector tab highlighted below, which allows you to view and interpret data in different formats.

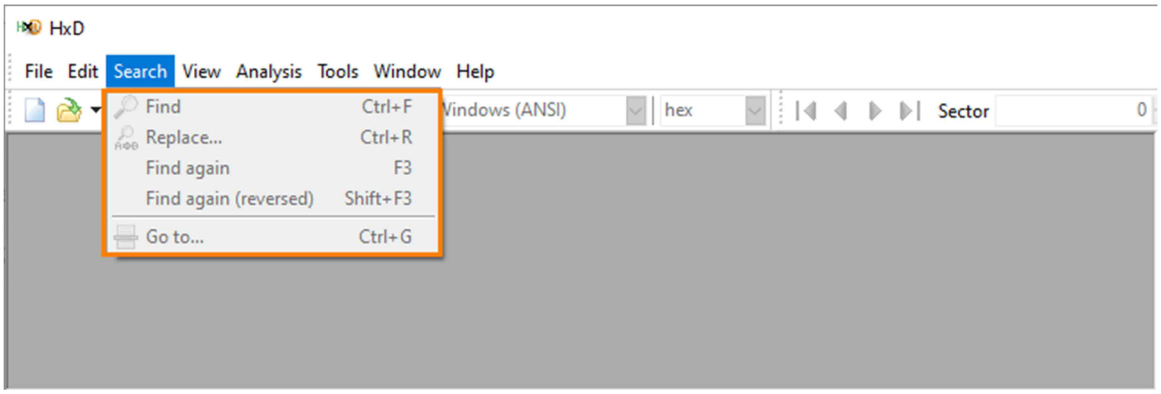


4. This powerful hex editor has many features and capabilities. We will cover the options that we will be using in the table below the following screenshots. The options we will be using are found in the Search, View, and Tools dropdown menus.

Tools Menu

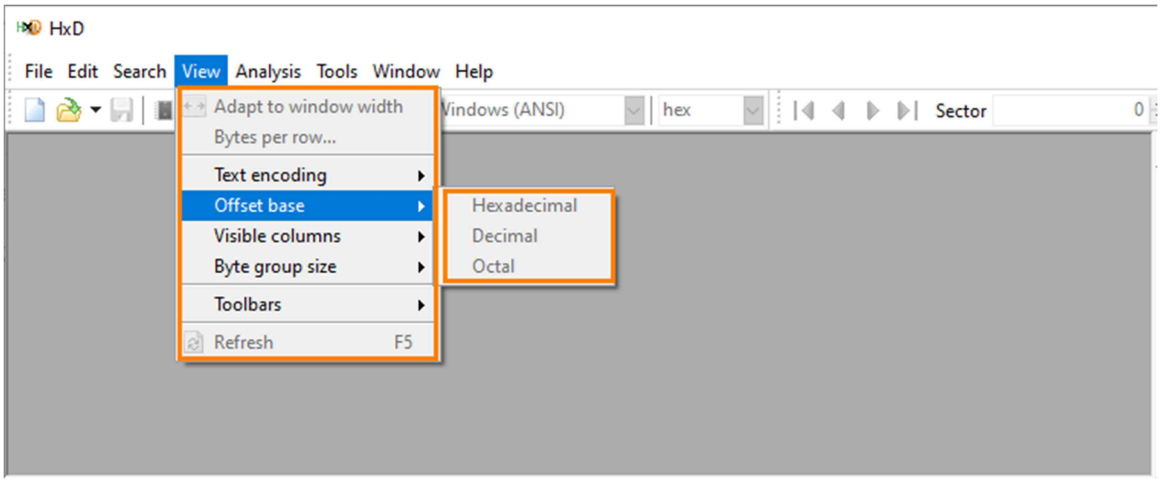


Search Menu



The options are grayed out until a file is added to the application that will enable the search function.

View Menu



5. As always, please note that all the options are not listed here. There are many other features that will not be used in this course. If you would like to learn more about the tool, feel free to visit the HxD website https://mh-nexus.de/en/hxd/.
6. Now that you are familiar with some basic features of HxD, let us use it to look at some FEFs.

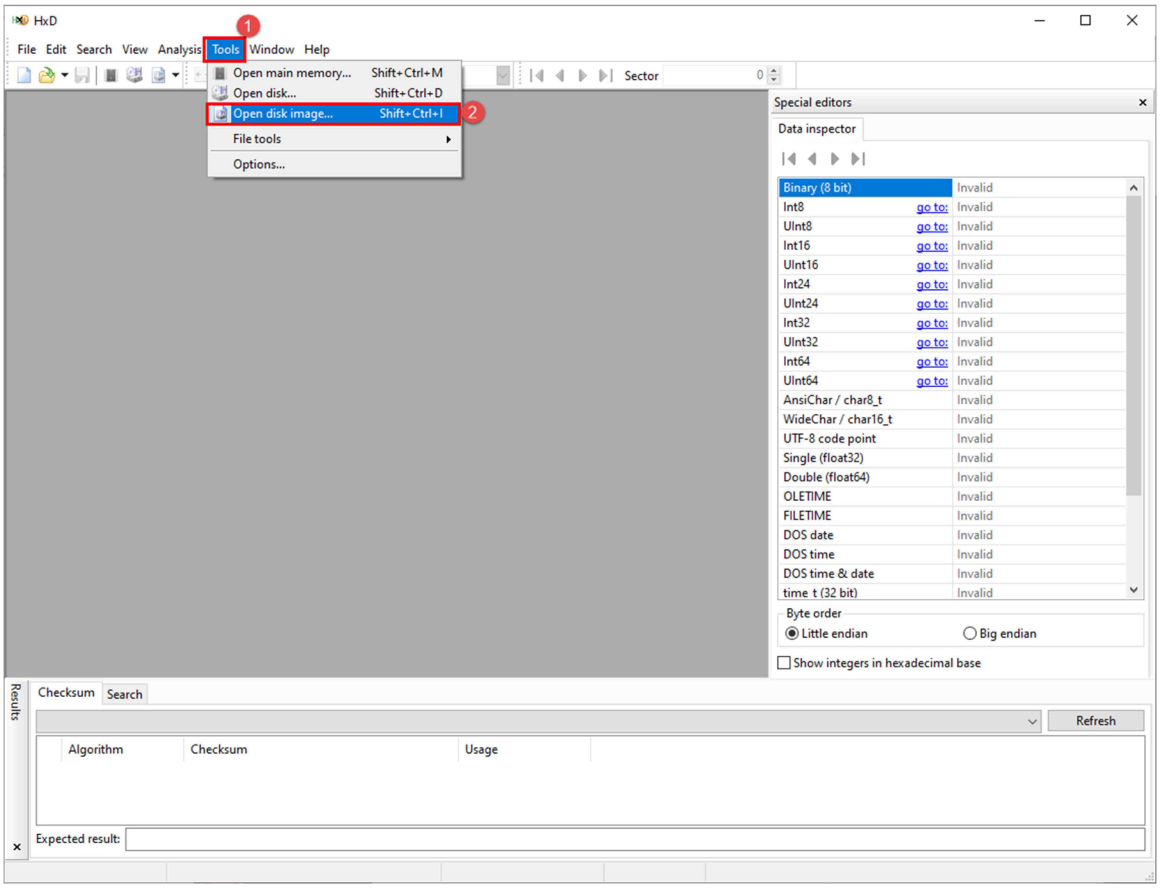
Open disk	The *Open disk* option allows you to add a local disk and view it in hexadecimal and raw text view.
Open disk image	The *Open disk image* option allows you to open a disk image file and view it in hexadecimal and raw text view.
Find	The *Find* option allows you to search the data that is shown in the main view pane.
Go to	The *Go to* option allows the user to go to a specific Offset in the data that is shown in the main view pane.
Offset base	The *Offset base* option allows you to switch between Hex, Decimal, and Octal characters in the view pane.

2 Identifying File System Data in a FAT Formatted Evidence File

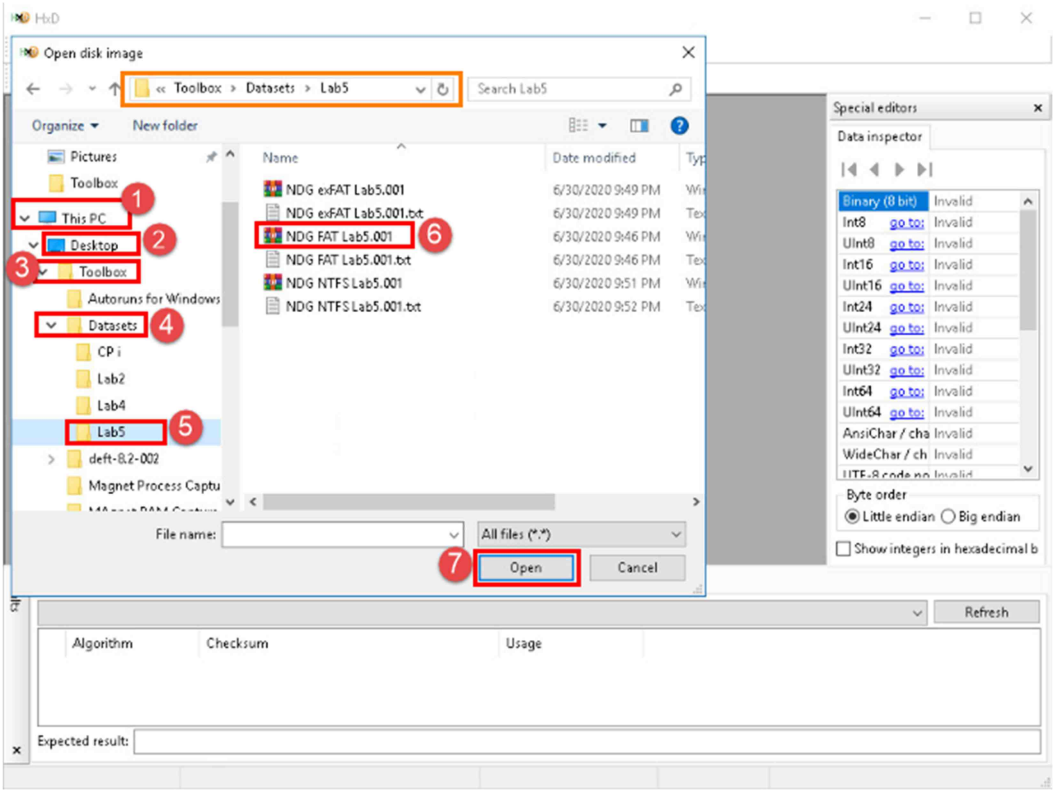
All data stored on a storage device (hard drive or Solid-State Drive) is in the form of a magnetic field or electric charge and can be referred to as a binary digit or bit. To allow access to storage devices, there needs to some form of structure. This is done by the computer and is referred to as a Logical Disk Structure (Partitions and File Systems). To write data to the hard drive, the Logical Disk Structure allocates areas of the drive into individual blocks. This process must first be completed before the drive is usable.

In this lab, we will be reviewing how each Partition and File System is formatted.

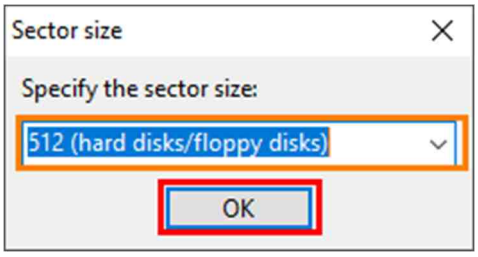
1. Let us use HxD to review the FEF and learn how to read the data contained in a partition table. You should still have HxD open. If not, reopen it and click the Open disk image option from the Tools dropdown menu, as seen in items 1 and 2 in the screenshot below.



2. The Open disk image window will appear. Use this window to browse to This PC > Desktop and double-click the folder Toolbox > Datasets > Lab5. This will open the folder revealing 3 FEFs. Select the file called NDG FAT Lab5.001 and click the Open button as highlighted below.

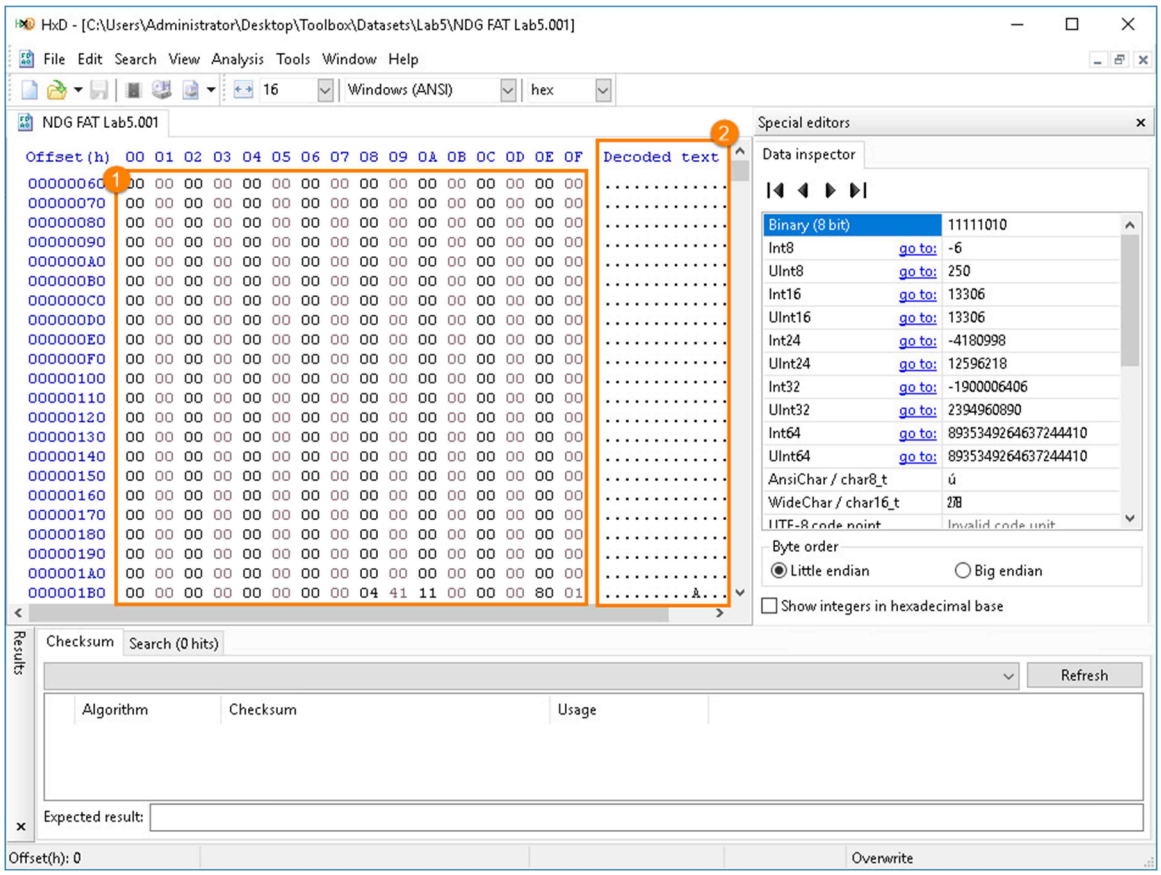


3. The Sector size window will appear. This window allows you to select the sector size of the image. In this lab, we will leave the option as 512 (hard disks/floppy disks) and click OK as highlighted below.



Do NOT change the specified sector size from 512 Bytes.

4. You will see the window below appear. As you can see in the screenshot below, the view pane now contains the hexadecimal representation on the left of the pane. This is highlighted as item 1 below. Immediately beside the hexadecimal values is the Decoded text view, highlighted as item 2.



5. Now let us begin reviewing the Master Boot Record (MBR – Partition Table). The MBR - partition table can be found in the first sector (sector 0) of storage disks. It can be broken into 3 sections: the Bootstrap code Area, the partition table, and the Boot Record Signature. Each of these sections is stored at specific offsets. The table below provides a description of their locations and sizes.

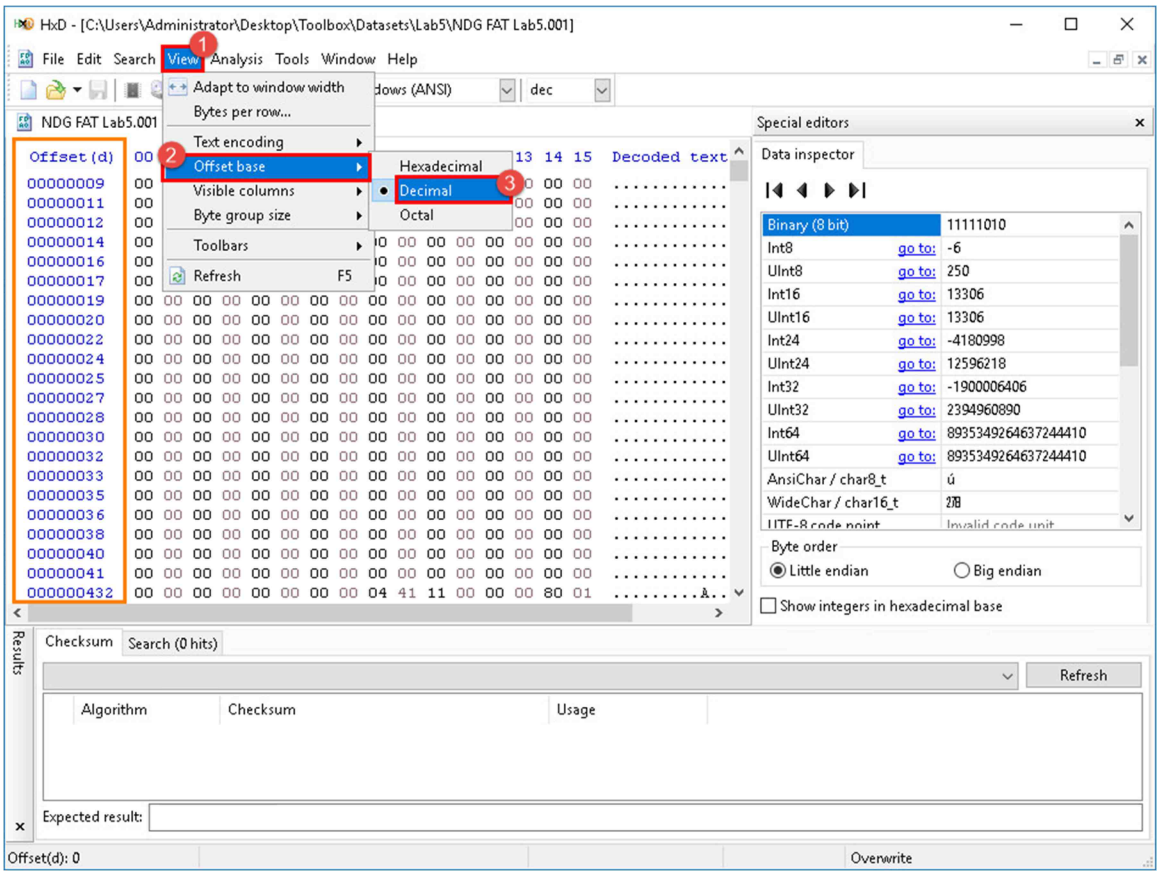
Structure of a generic MBR			
Offsets within sector		Length (in Bytes)	Description
Decimal	Hexadecimal		
000 - 445	000 – 1BD	446	Bootstrap Code Area
446 - 509	1BE – 1FD	64	Partition Table
510 - 512	1FE – 1FF	2	Boot Record Signature

http://blog.hakzone.info/posts-and-articles/bios/analysing-the-master-boot-record-mbr-with-a-hex-editor-hex-workshop/

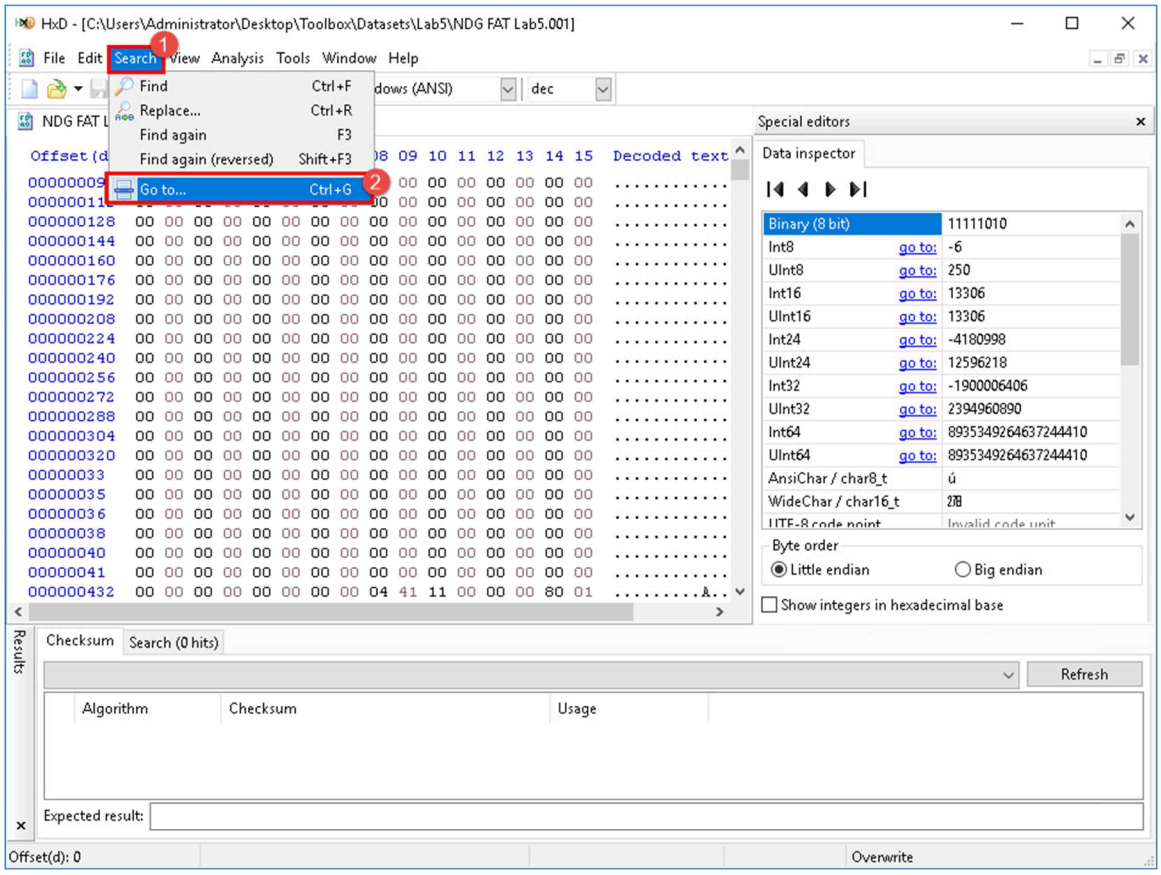
6. As you saw in the table above, the partition table is located at offset 4446 – 509. We will be focusing on the data within the partition table in this exercise. Each partition entry in the partition table is 16 bytes long. Typically, each drive can have 4 primary partitions or 3 primary partitions and 1 extended partition. This makes sense since the partition table is only 64 bytes in size (446 - 509 bytes); it can only store 4 entries, and each entry is 16 bytes long.

The additional extended partition allows for more logical partitions to be created; however, extended partitions will not be covered in this lab.

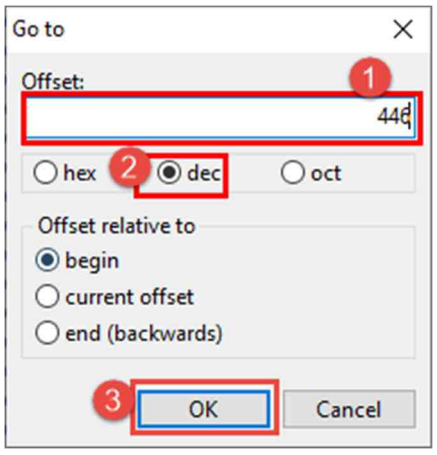
7. Now, let us look at the data. Since we will be using decimal values to go to offset 446, we will need to change the Offset base to decimal. To do this, click the View dropdown menu option from the menu bar and hover over the Offset base option, then select Decimal as highlighted in items 1, 2, and 3 below.



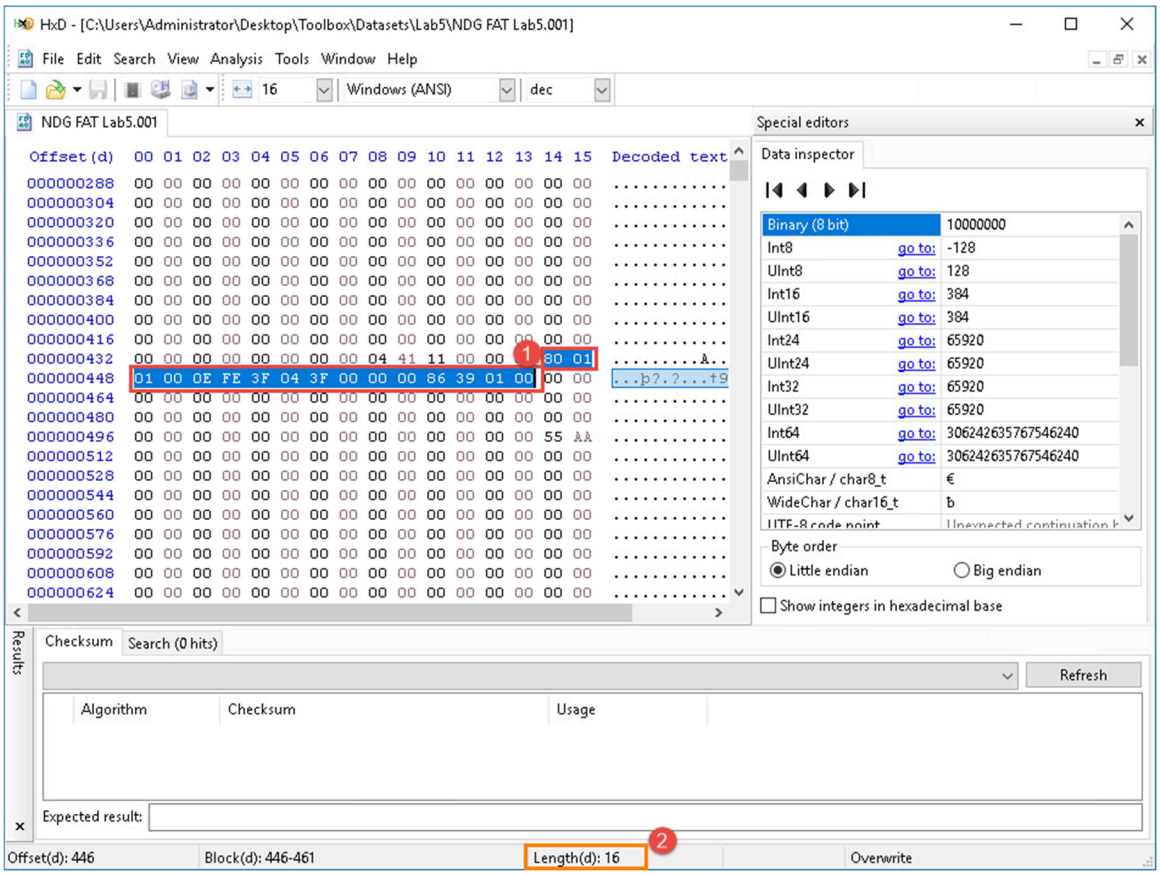
8. Now that the offsets are in decimal, let us go to offset 446. Once again, go to the menu bar. This time click the Search dropdown menu option and then click Go to from the dropdown menu or press Ctrl+G as highlighted in items 1 and 2 below.



9. The Go to window will appear. This window allows you to enter an offset, and it will place the cursor at the beginning of the offset. Let us type 446 in the text box highlighted as item 1 below. Next, click the radio button beside dec, as seen in item 2 below. This tells Go to that you are searching for a decimal value. Once you have verified that everything is correct, leave the other option as default and click OK as seen in item 3 below.



10. Your cursor will be taken to offset 446. Let us highlight the 16 bytes after the cursor, as seen in item 1 below. You can use the status bar at the bottom of the main window to count the length of your selection, as highlighted in item 2 below.



11. The data you just highlighted is the partition entry for the first partition on the disk. This highlighted data can be broken up into 6 sections. The table below shows the different sections.

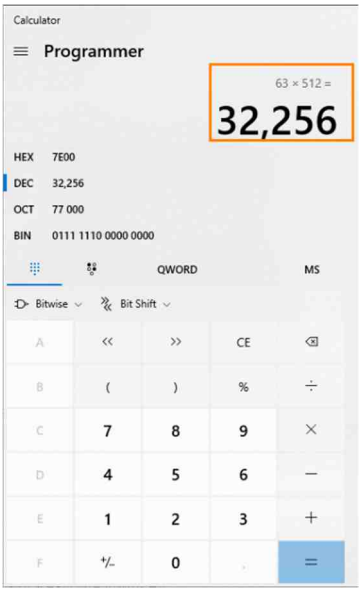
    0x80 | 0x01 01 00 | 0x0E | 0xFE 3F 04 | 0x3F 00 00 00 | 0x86 39 01 00

Offset	Length (in Bytes)	Description
0x80	1	This is the first character and denotes whether the partition is active or not. 0x80h indicates that the partition is active. Alternately, 0x00 would indicate that the partition is inactive. (0 = Non-Bootable / 80 = Bootable).
0x01 01 00	3	The next 3 bytes represent the starting sector of the partition. It is stored as a Cylinder Head Sector (CHS) value and is also in little-endian. This means the starting sector is 0x00 01 01.
0x0E	1	The fifth (5th) value represents the type of filesystem that is on this partition. *0E* represents a *FAT* file system.
0xFE 3F 04	3	The next 3 bytes represent the ending sector of the partition. It is also stored as a Cylinder Head Sector (CHS) value and is in little-endian. This means the partition's ending sector is located at 0x04 3F FE.
0x3F 00 00 00	4	The next 4 values indicate the starting sector of the file system in hexadecimal. It is stored in little-endian and so the value is 0x00 00 00 3F or just 0x3F. You can highlight the characters and view the converted data in the Data inspector pane. 0x3F converted to decimal is 63, which indicates that the starting sector for this file system is sector 63.
0x86 39 01 00	4	The last 4 values represent the number of sectors in the partition. This too is stored in little-endian so should be viewed as 00 01 39 86. You can highlight the characters and view the converted data in the *Data inspector* pane. When you convert this value to decimal, you will get 80262 sectors. To get the partition size, multiply the number of sectors (80262) by the size of each sector (512), in this case, 40,094,144 bytes (approx. 40MB.).

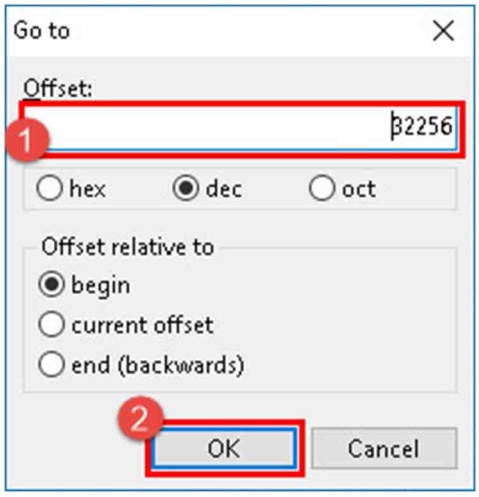
12. The partition entries precede the Boot Record Signature; this can be found at the end of the MBR as hex 55 AA (0x55AA) (bytes 510 and 511).

Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
000000288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000368	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000384	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000400	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000416	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000432	00	00	00	00	00	00	00	00	04	41	11	00	00	00	80	01
000000448	01	00	0E	FE	3F	04	3F	00	00	00	86	39	01	00	00	00
000000464	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000496	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA

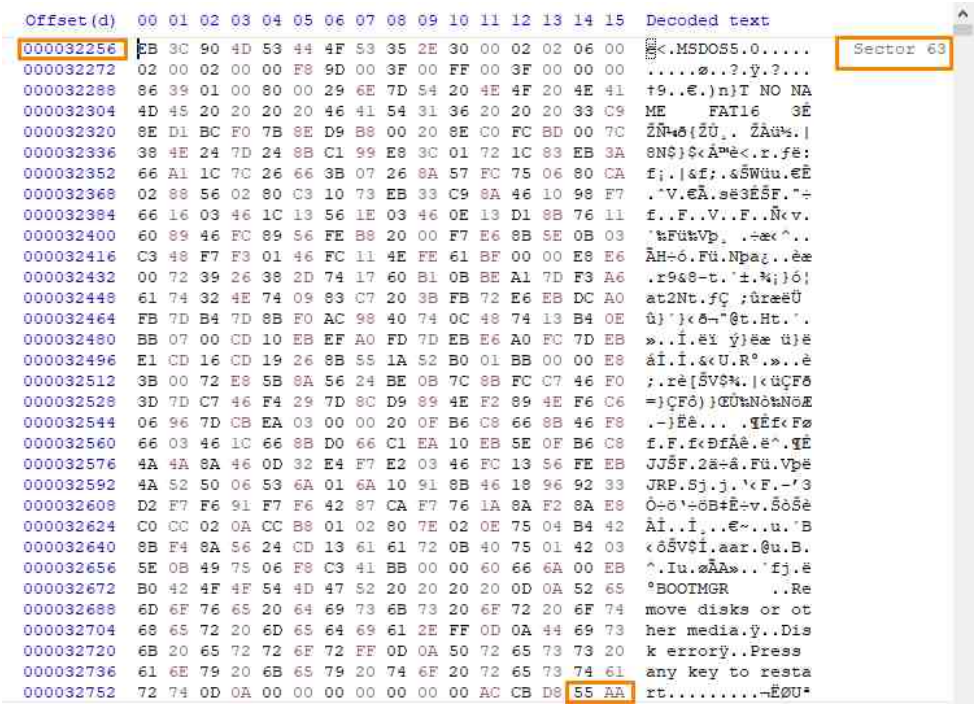
13. Now that we learned how to read the partition table, let us move on to the file system. We learned from the partition table that the starting sector for the file system is sector 63. Since each sector is 512 bytes, let us multiply 512 by 63 to get the offset. The result should be 32256.



14. As we did earlier, open the Go to window by clicking the Search dropdown menu option and then clicking Go to from the dropdown menu or press Ctrl+G. Once the window appears, type `32256` in the text box as highlighted as item 1 below. Verify that the radio button beside dec is still selected, and then click OK as seen in item 2 below.



15. Now that you are at sector 63, you will be looking at the text and hexadecimal representation of the volume boot record (VBR). The offset 32256 is the location of the first byte in sector 63 (VBR) on this volume. Since we know that each sector is 512 bytes, we can determine that the sector ends at offset 32767 by adding the number of bytes after the first byte (511) and the offset of the first byte (32256). Certain artifacts are located at specific byte offsets within the VBR.

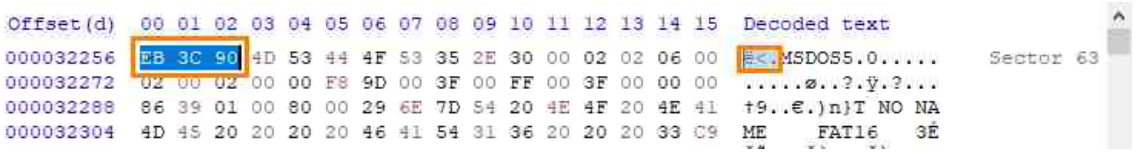


16. The table below provides the location of the artifacts based on their offset values for the entire volume as well as for the sector. During this exercise, we will use data from the Dec and Universal Offsets columns to locate the relevant entries in the VBR located at sector 63.

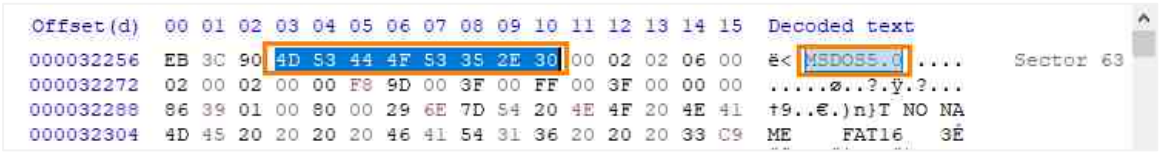
Offset		Universal Offsets		Length (Bytes)	Name	Description
Hex	Dec					
0x7E00	32256	0x00	0	3	Jump Instruction	Jump instructions to skip to boot code field
0x7E03	32259	0x03	3	8	OEM ID	ASCII - MSDOS5.0
0x7E0B	32267	0x0B	11	2	Bytes per Sector	Combined will provide Cluster size
0x7E0D	32269	0x0D	13	1	Sectors per Cluster	
0x7E20	32288	0x20	20	8	Total Sectors on Volume	Volume size
0x7E15	32277	0x15	21	1	Media Descriptor	Common values are 0xF8 and 0xF0 which represents fixed media and removable media, respectively
0x7E27	32295	0x27	39	4	Volume Serial Number	Serial number of the Volume
0x7E36	32310	0x36	54	8	FAT16	File System Type
0x7FFE	32766	0x1FE	510	2	Boot Signature	0x55 AA

Normally, the universal offsets would be used to find each section within the Sector 63, however, HxD will display the entire physical disk which will contain sector 63.

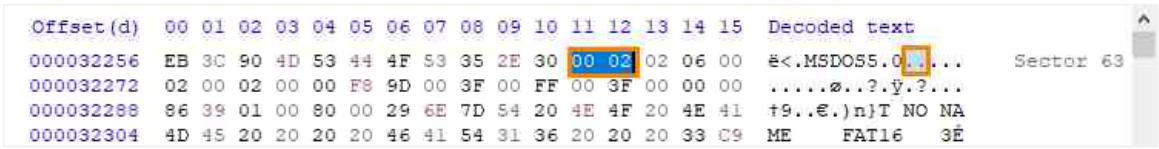
17. Let us highlight the first 3 bytes starting at offset 32256. These bytes are known as the Jump instruction, and it instructs the computer to skip over the next few bytes as they are not executable.



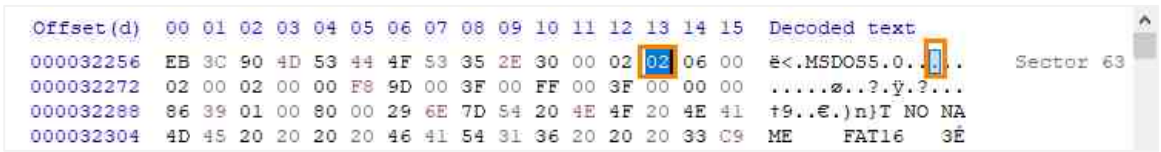
18. Highlight the next 8 bytes immediately after the Jump instruction. These bytes are located at offset 32259 in our FEF or the 3rd byte from the beginning of the sector. These 8 bytes are the OEM ID and will tell you the name of the file system when converted to text. As seen in the screenshot below, the highlighted text is MSDOS5.0, which indicates that it is a FAT file system.



19. Let us highlight the next 2 bytes after the OEM name as indicated below. These bytes are located at offset 32267 in our FEF or the 11th byte from the beginning of the sector. The highlighted bytes are 0x00 02, and these bytes indicate the number of bytes per sector. When converted to decimal, the number of bytes is 512.

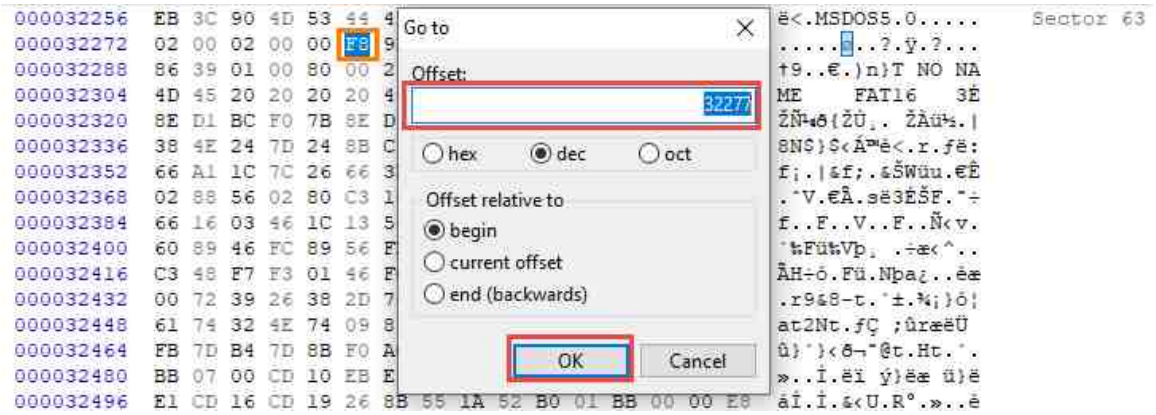


20. Now highlight the byte immediately beside the previous one as indicated below. This byte is located at offset 32269 in our FEF or the 13th byte from the beginning of the sector. This value is the number of sectors per cluster and, as seen below, is 0x02 or 2 in decimal. Combining this value with the number of bytes per sector (512) can provide the cluster size for the volume.

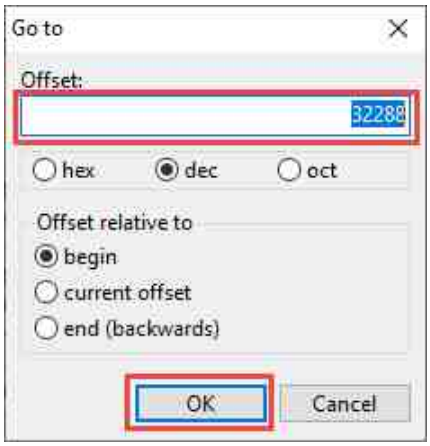


In this case, 512 bytes per sector * 2 sectors = size of each cluster 1024 bytes.

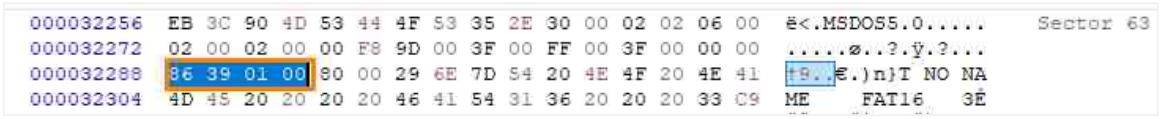
21. Now that we know the size per cluster, let us look at the media type. This can tell whether the volume is a removable/fixed disk, a floppy disk, or other type of medium. The media type is located at offset 32277 in our FEF or the 21st byte from the beginning of the sector. Let us use Go to again; it should be open already; if not, reopen it and type `32277` and click OK. This will take you to the beginning of the media type entry. Highlight the next byte, as seen below. The hex value F8 represents a hard disk or removable drive (USB drive etc.).



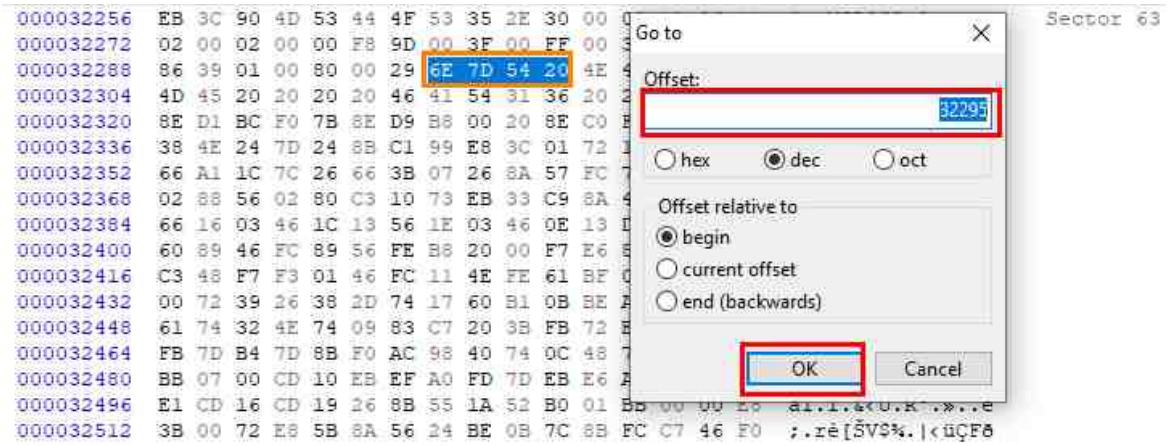
22. Let us jump to offset 32288 or the 32nd byte from the beginning of the sector. You will find the total number of sectors on the volume here. Let us use Go to locate the value. Reopen it by clicking the Go to option from the Search dropdown menu option on the menu bar or using Ctrl+G. In the Go to window, type `32288` and click OK. You will be taken to the offset 32288.



23. Let us highlight the next 4 bytes, as seen below. This value is represented as 0x86 39 01, which when converted to decimal is 80262. This indicates that there are 80262 sectors. With this information, we can determine the size of the volume. The size can be found by multiplying the number of sectors (80262) by the sector size (512), which is equal to 41,094,144 bytes (41MB).

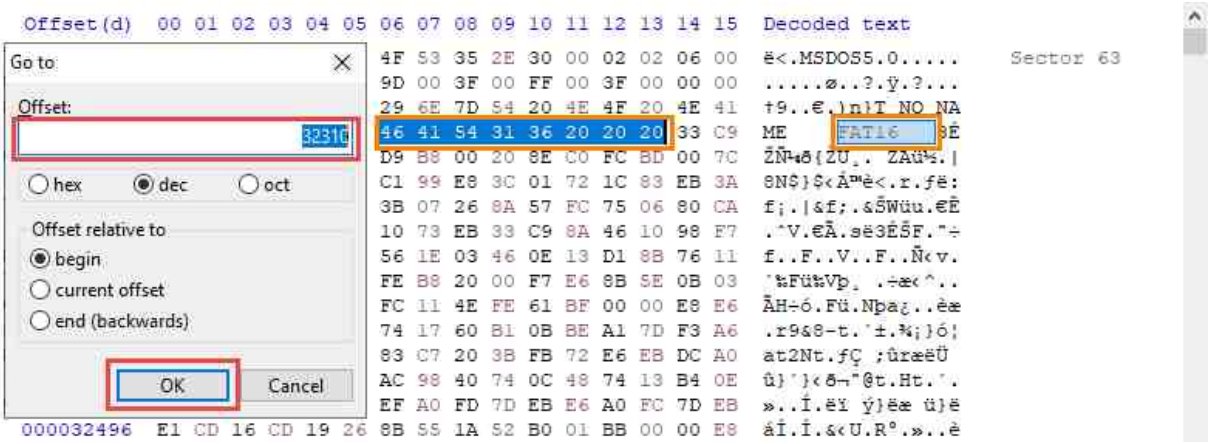


24. Next, let us look at the volume serial number. This can be found at the 39th, byte from the beginning of the sector, or offset 32295 in our FEF. Use Go to by browsing to the Search dropdown menu on the menu bar and clicking Go to or using Ctrl+G. Type 32295 and click OK. Highlight the next 4 bytes, as seen below.



The volume serial number is created when the drive is formatted and can be used to determine if files were ever stored on the drive. The Volume Serial Number byte order is in little-endian, which means it is read from right to left 2054-7D6E.

25. Let us look at the Filesystem type. As the name suggests, this artifact will tell what type of filesystem is on the volume. The Filesystem type is located at offset 32310 in our FEF or the 54th byte from the beginning of the sector. Let us use Go to again; it should be open already; if not, reopen it and type `32310` and click OK. This will take you to the beginning of the Filesystem type entry. Highlight the next 8 bytes, as seen below. As you can see in the textual version of the highlighted text, the Filesystem type is FAT16.

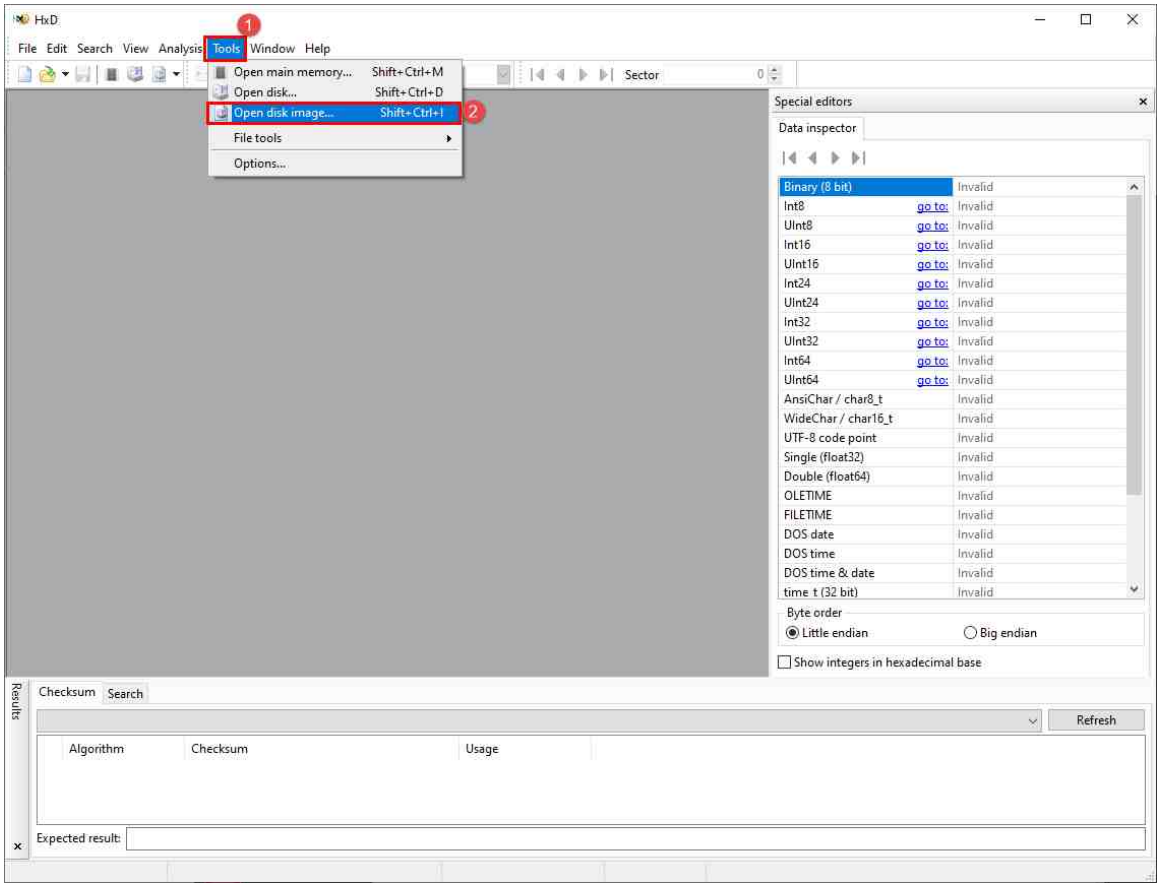


26. The artifacts we covered were only some of the things that are stored in the FAT file system. Each version of the file system stores data at a slightly different location. When manually reviewing the MBR and VBR as we just did, be sure to use a chart or guide to identify the necessary artifacts associated with the file system you are examining.
27. Let us move on to another type of file system.

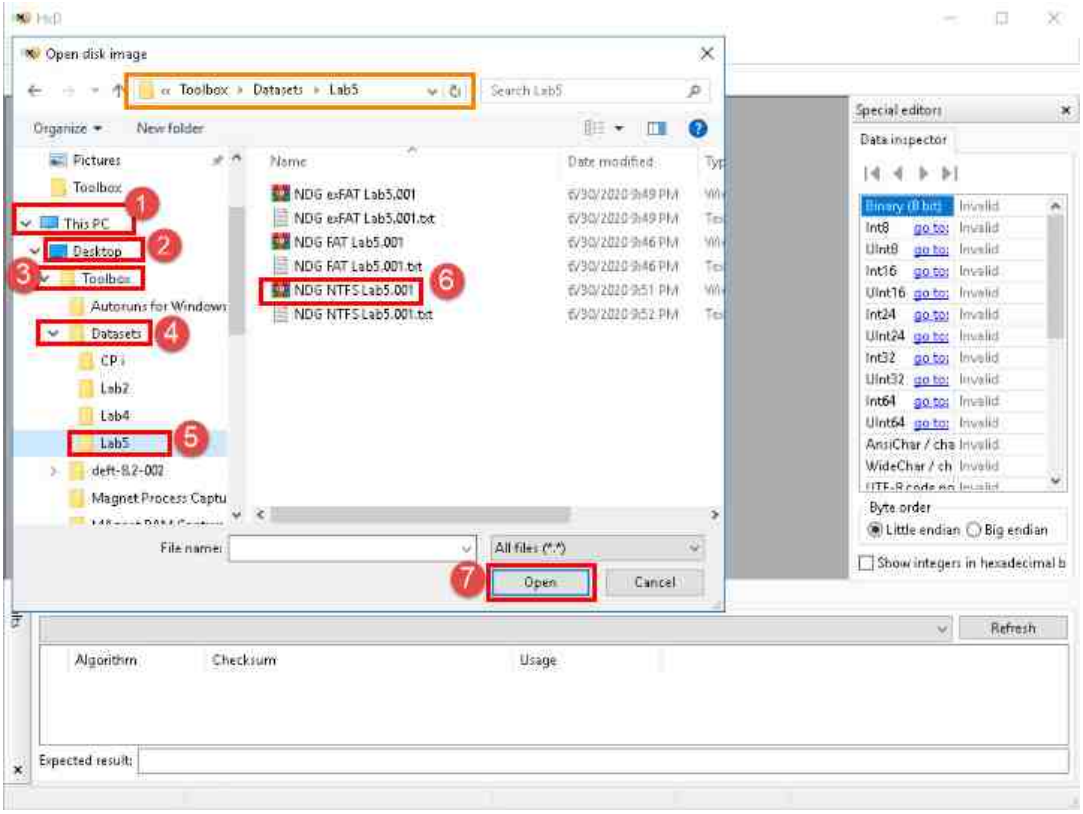
3 Identifying File System Data in an NTFS Formatted Evidence File

NTFS stands for New Technology File System and is a file system created by Microsoft; it is the file system used with most versions of Microsoft Windows. In this exercise, we will review the artifacts you can find in the NTFS VBR.

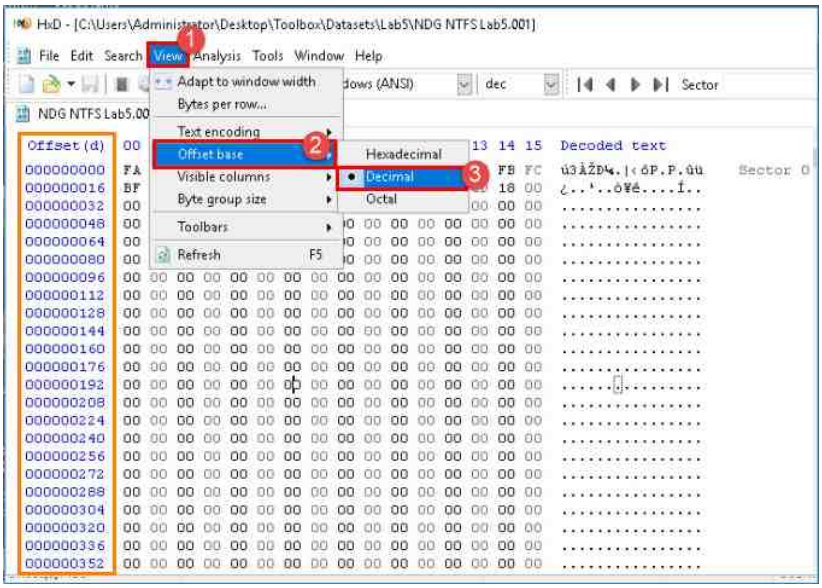
1. Let us use HxD to review the FEF and learn how to read the data in the partition table. You should still have HxD open; if not, reopen it and click the Open disk image option from the Tools dropdown menu, as seen in items 1 and 2 in the screenshot below.



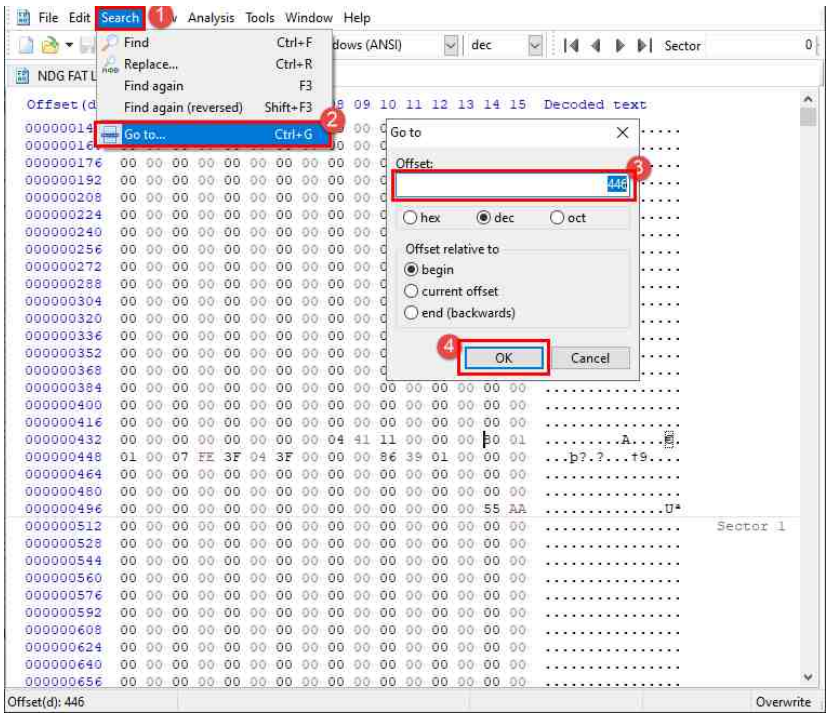
2. The Open disk image window will appear. Use this window to browse to This PC > Desktop and double-click the folder Toolbox > Datasets > Lab5. This will open the folder revealing 3 FEFs. Select the file called NDG NTFS Lab5.001 and click the Open button as highlighted below. The NTFS image file will now be loaded in HxD.



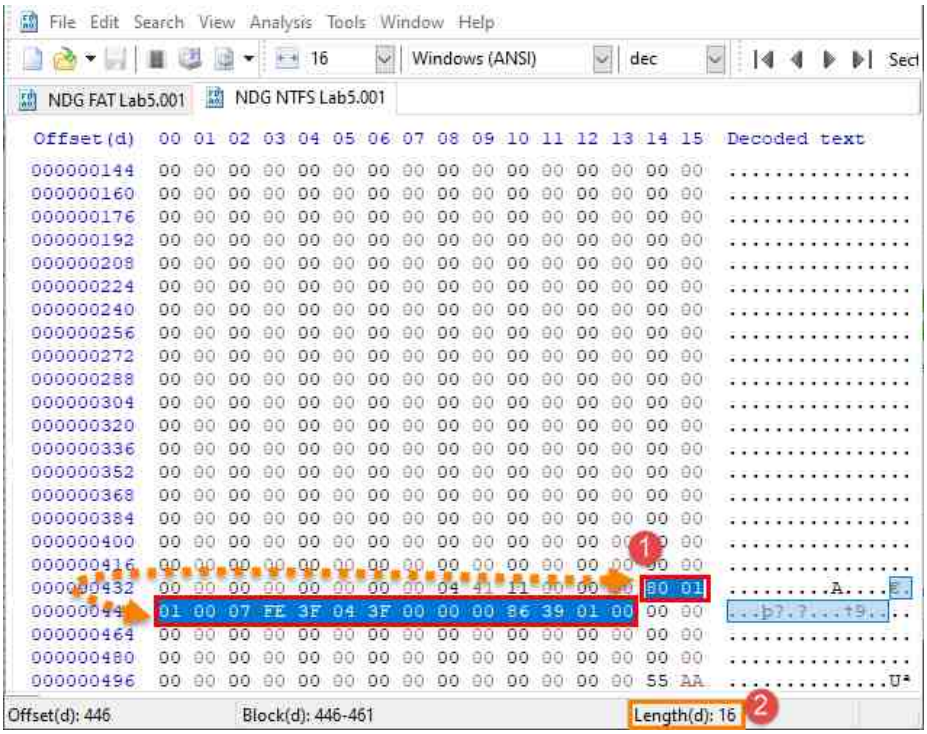
3. As we did before, let us begin by reviewing the Master Boot Record (MBR – Partition Table) in the first sector to locate the VBR. As we learned in the previous exercise, the first partition entry is located at offset 446. Since we will be using decimal to go to offset 446, we will need to change the Offset base to decimal. To do this, click the View dropdown menu option from the menu bar and hover over the Offset base option, then select Decimal as highlighted in items 1, 2, and 3 below.



4. Now that the offsets are in decimal, let us use Go to Ctlg jump to offset 446. This can be opened by going to the Search dropdown menu option Ctrl+G in the menu bar and then clicking Go to from the dropdown menu or pressing Ctrl+G. When the Go to window appears, type 446 in the text box, verify that dec is selected, and then click OK as seen below.



5. Your cursor will be taken to offset 446. Let us highlight the 16 bytes after the cursor, as seen in item 1 below. You can use the status bar at the bottom of the main window to count the length of your selection, as highlighted in item 2 below.

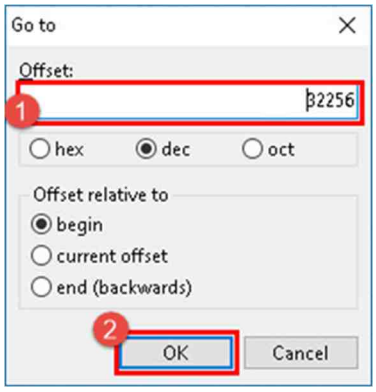


6. As we did with the partition entry above, let us break it down to learn what type of file system the entry is referring to and what sector the VBR is located in.

The highlighted value is 0x80 | 0x01 01 00 | 0x07 | 0xFE 3F 04 | 0x3F 00 00 00 | 0x86 39 01 00

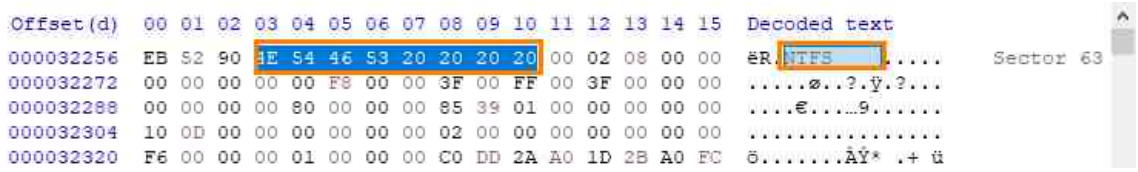
7. The only difference between this entry and the previous FAT entry is that the 5th byte that represents file system type is 07, which is used to represent NTFS/exFAT file systems. The starting sector of this file system (where the VBR is located) is the same as before. When converted, it is 63. Let us jump to sector 63.
8. Since we know that each sector is 512 bytes, we can determine that the sector ends at offset 32,767 by adding the number of bytes after the first byte (511) and the offset of the first byte (32256). Certain artifacts are located at specific byte offsets within the VBR. Like before, we will provide the location of the artifacts based on their offset values for the entire volume as well as for the sector.
9. The table below provides the location of the artifacts based on their offset values for the entire volume as well as for the sector. During this exercise, we will use data from the Dec and Universal Offsets columns to locate the relevant entries in the VBR located at sector 63.

10. Since the calculations for this file system are the same as the FAT file system we reviewed earlier, we already know that the starting sector is located at offset 32256 in our FEF. Let us reopen Go to by clicking the Search dropdown menu option, and clicking Go to from the menu, or typing Ctrl+G. In the Go to window, type `32256` and click OK as highlighted below.

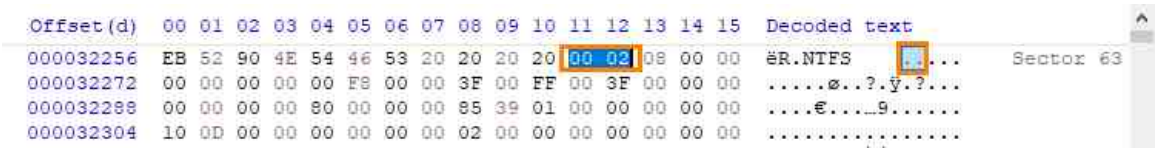


Offset		Universal Offsets		Length (Bytes)	Name	Description
Hex	Dec					
0x7E00	32256	0x00	0	3	Jump Instruction	Jump instructions to skip to boot code field
0x7E03	32259	0x03	3	8	OEM ID	ASCII – NTFS
0x7E0B	32267	0x0B	11	2	Bytes per Sector	Combined will provide Cluster size
0x7E0D	32269	0x0D	13	1	Sectors per Cluster	
0x7E20	32288	0x28	40	8	Total Sectors on Volume	Volume size
0x7E15	32277	0x15	21	1	Media Descriptor	Common values are 0xF8 and 0xF0 which represents fixed media and removable media, respectively
0x7E48	32328	0x48	72	4	Volume Serial Number	Serial number of the Volume
0x7FFE	32766	0x1FE	510	2	Boot Signature	0x55 AA

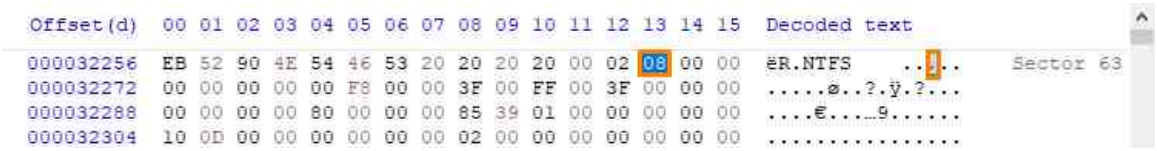
11. At offset 32256 (Sector 63), begin by highlighting the OEM ID. This is located at offset 32259 in our FEF or the 3rd byte from the beginning of the sector. Highlight the following 8 bytes after 32259 to reveal the OEM ID as highlighted below. As seen in the textual version of the highlighted data, the OEM ID is NTFS, which indicates that it is an NTFS file system.



12. Highlight the next 2 bytes that follow the OEM ID as highlighted below. These are located at offset 32267 in our FEF or the 11th byte from the beginning of the sector. As seen below, the value is 0x00 02 or 512 bytes when converted to decimal and indicates the number of bytes per sector.

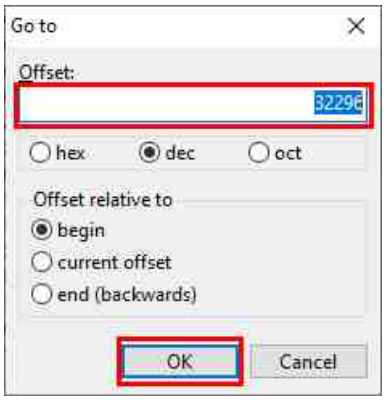


13. Now highlight the byte immediately beside the previous one, as seen below. This byte is located at offset 32269 in our FEF or the 13th byte from the beginning of the sector. This value is the number of sectors per cluster and, as seen below, is 0x08 or 8 in decimal. Combining this value with the number of bytes per sector can provide the cluster size for the volume.

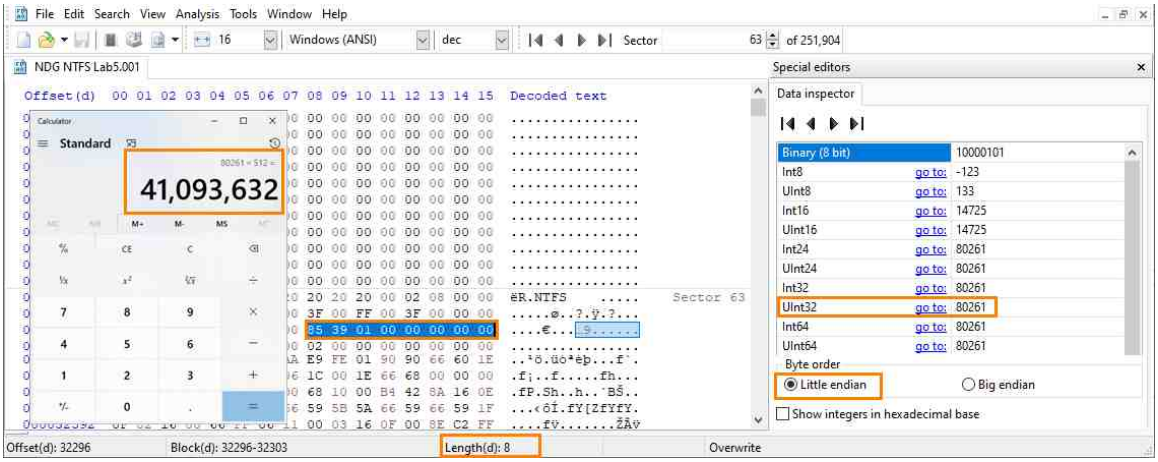


In this case, 512 bytes per sector * 8 sectors = size of each cluster 4096 bytes.

14. Let us jump to offset 32296 or the 40th byte from the beginning of the sector. In an NTFS VBR, you will find the total number of sectors on the volume here. Let us use Go to, in order to locate the value. Reopen it by clicking the Go to option from the Search dropdown menu option on the menu bar. In the Go to window, type `32296` and click OK. You will be taken to the offset 32296.

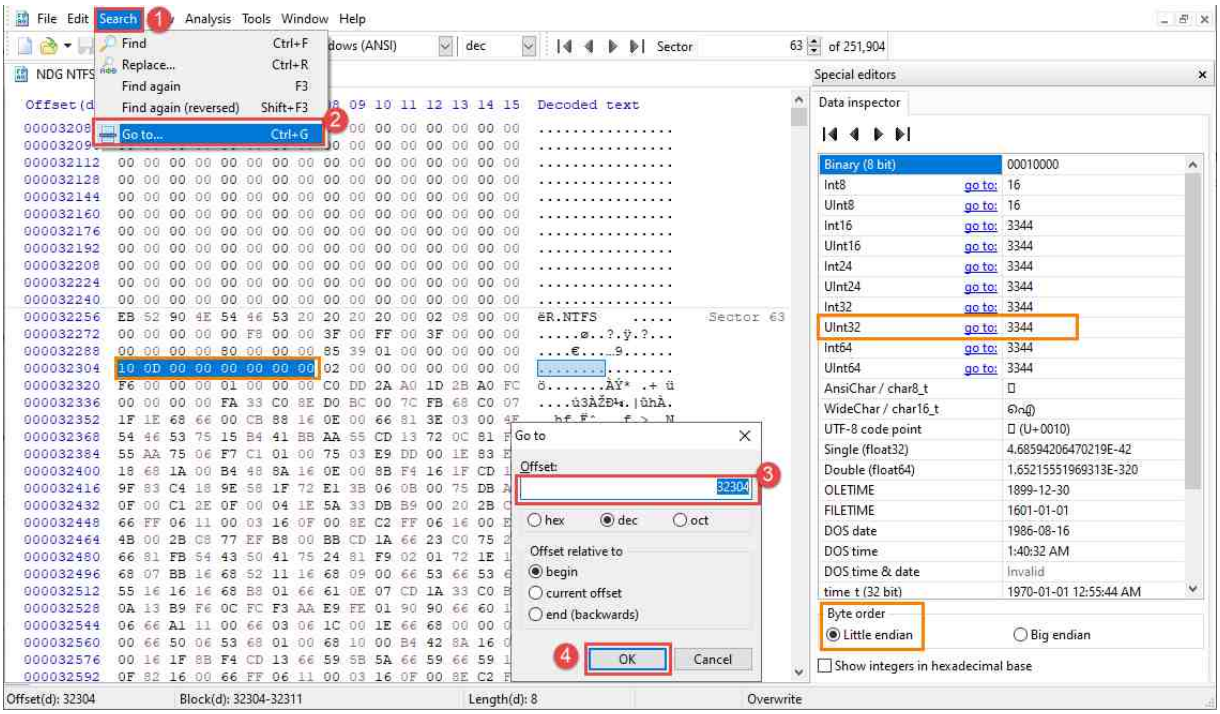


15. Let us highlight the next 8 bytes as highlighted below. This value is represented as 0x85 39 01 00 00 00 00 00, which when converted to decimal is 80261. This indicates that there are 80261 sectors in the volume. The size of the volume in bytes can be found by multiplying the number of sectors (80261) by the sector size (512), which is equal to 41,093,632 bytes (41MB).

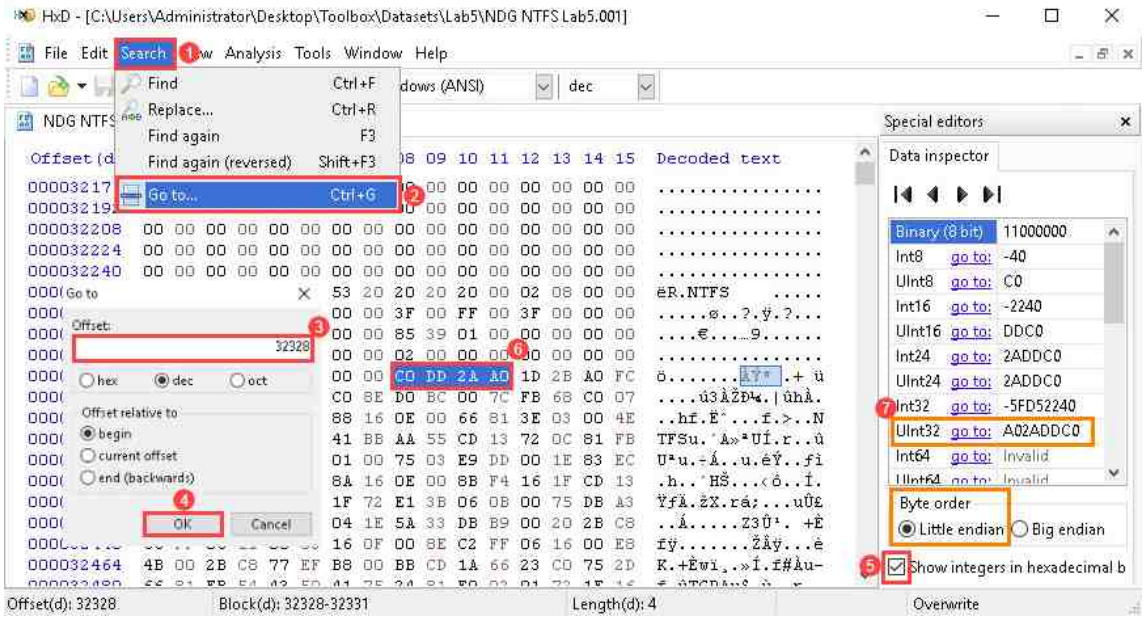


The byte order of the values represented as 0x85 39 01 00 00 00 00 00 is little-endian, which mean the values are reversed 0x00 00 00 00 01 39 85. The preceding zeroes are dropped leaving 0x01 39 85 which converts to 80261.

16. Next, let us identify the starting extent for the Master File Table (MFT). This is located at offset 32304 in our FEF or the 48th byte from the start of the sector. Let us use Go to by browsing to the Search dropdown menu on the menu bar and clicking Go to or using Ctrl+G. Type `32304` and click OK. Highlight the following 8 bytes as seen below. These bytes are 0x10 0D 00 00 00 00 00 00 and represent 3344 when converted to decimal. This number is the starting cluster of the MFT. An examiner would be able to browse to the specific cluster and extract or view the contents of the MFT with this information.



17. Next, let us look at the volume serial number. This can be found at the 72nd byte from the beginning of the sector or offset 32328 in our FEF. Use Go to by browsing to the Search dropdown menu on the menu bar, and clicking Go to or using Ctrl+G seen in items 1 and 2. Type 32328 and click OK as seen in items 3 and 4. Before going any further, let us click the checkbox beside Show integers in hexadecimal base as seen in item 5. This will change the values in the Data Inspector tab to hexadecimal. Now that you are at the offset 32328 (752nd byte), highlight the next 4 bytes as seen in item 6 below. These bytes represent the volume serial number and are stored in little-endian. When converted, the volume serial number will read A02A-DDC0, as seen in item 7.

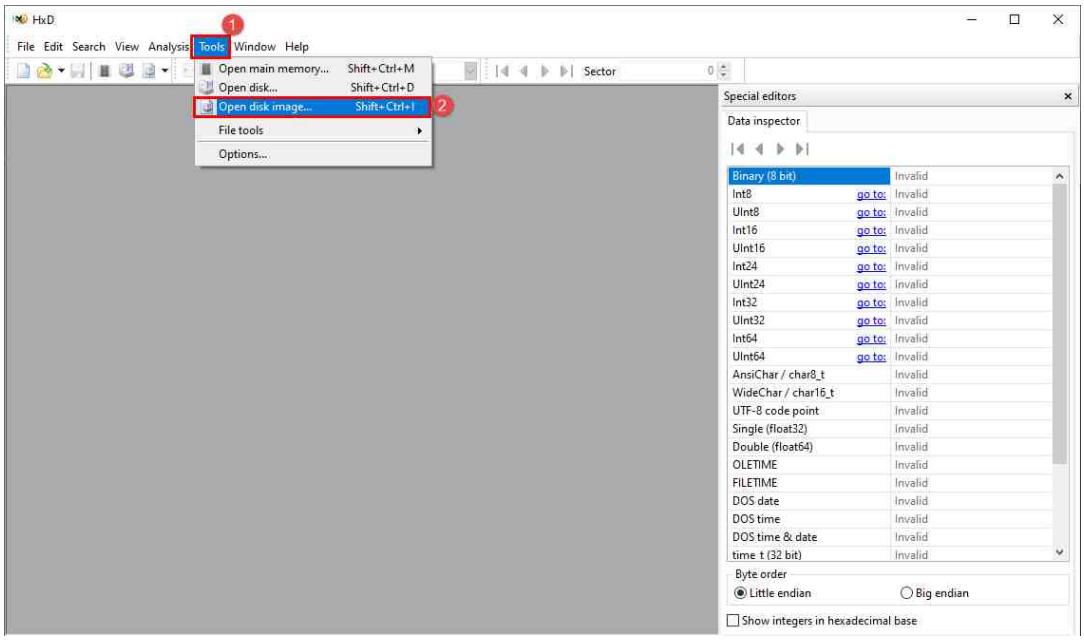


18. As with the previous file system, there are many artifacts that can be unearthed but will not be covered in this exercise. We will now review the final file system that this lab will cover.

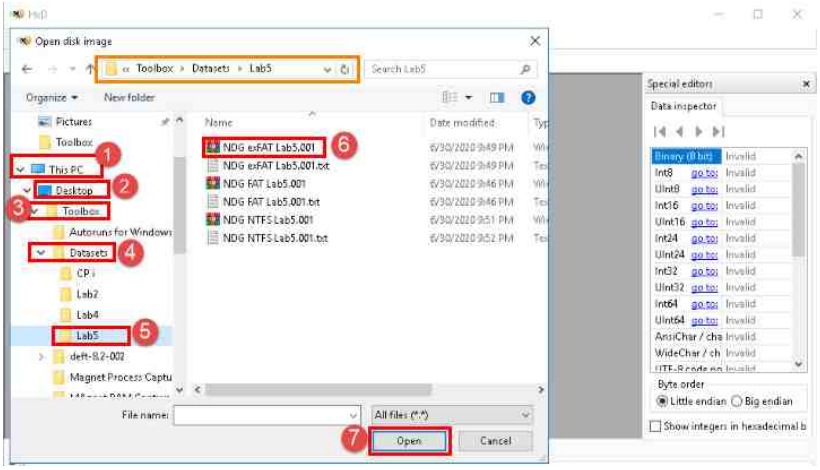
4 Identifying File System Data in an exFAT Formatted Evidence File

The Extensible File Allocation Table (exFAT) file system is becoming very common. It is like FAT but has a larger file capacity limit and can handle data better. As such, it is important to understand how to identify exFAT volumes and learn where to locate their file system artifacts. In this exercise, we will review the artifacts you can find in the exFAT VBR.

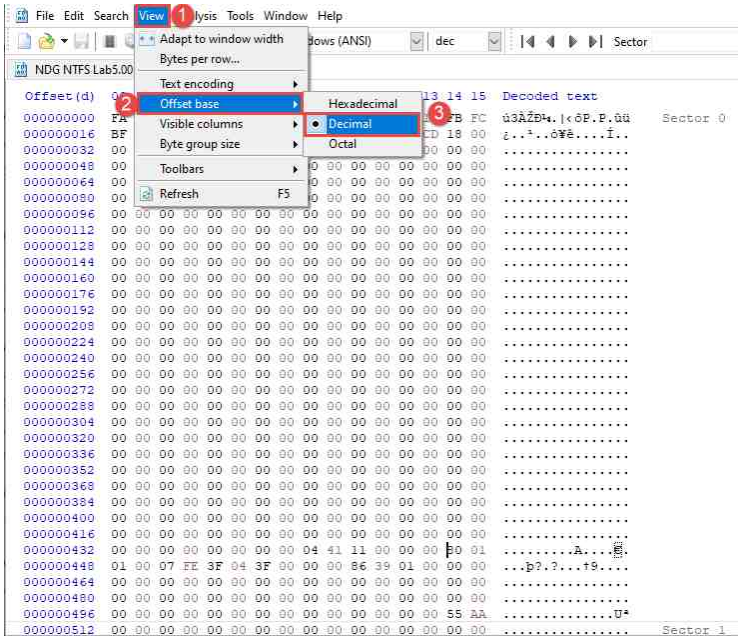
1. Let us use HxD to review the FEF and read the data stored in the partition table. You should still have HxD open. If not, reopen it and click the Open disk image option from the Tools dropdown menu, as seen in items 1 and 2 in the screenshot below.



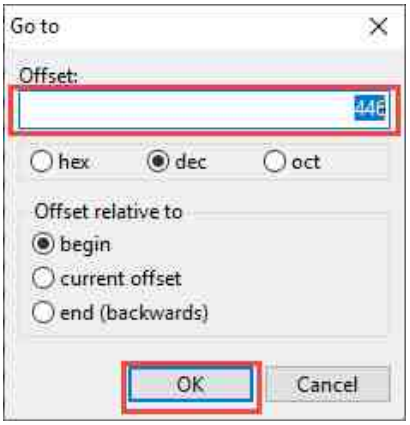
2. The Open disk image window will appear. Use this window to browse to This PC > Desktop and double-click the folder Toolbox > Datasets > Lab5. This will open the folder revealing 3 FEFs. Select the file called NDG exFAT Lab5.001 and click the Open button as highlighted below. The NTFS image file will now be loaded in HxD.



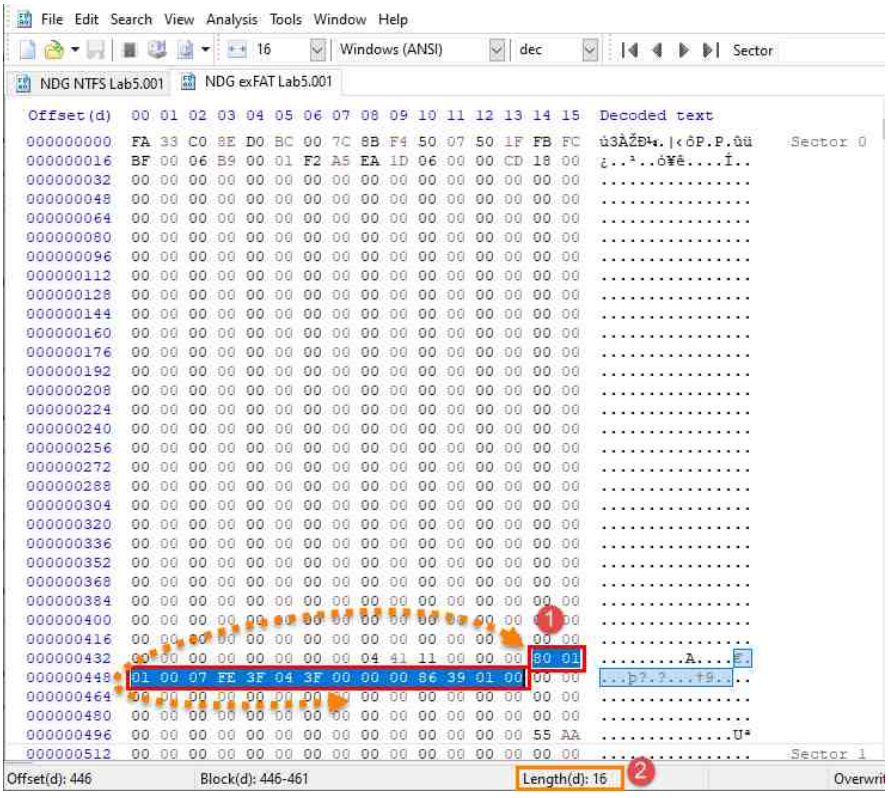
3. As we did before, let us begin by reviewing the Master Boot Record (MBR – Partition Table) in the first sector to locate the VBR. This means going to the first partition entry located at offset 446. Since we will be using decimal to go to offset 446, we will need to change the Offset base to decimal. To do this, click the View dropdown menu option from the menu bar and hover over the Offset base option, then select Decimal as highlighted in items 1, 2, and 3 below.



4. Now let us use Go to to jump to offset 446. This can be opened by going to the Search dropdown menu option in the menu bar and then clicking Go to from the dropdown menu or pressing Ctrl+G. When the Go to window appears, type `446` in the text box, verify that dec is selected, and then click OK as seen below.



5. Your cursor will be taken to offset 446. As we did before, highlight the 16 bytes after the cursor, as seen in item 1 below. You can use the status bar at the bottom of the main window to count the length of your selection, as highlighted in item 2 below.



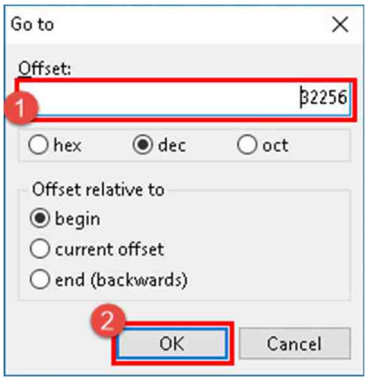
6. Let us break it down to learn what type of file system the entry is referring to and the location of the VBR.

The highlighted value is 0x80 | 0x01 01 00 | 0x07 | 0xFE 3F 04 | 0x3F 00 00 00 | 0x86 39 01 00

7. There is no difference between this entry and the NTFS entry because the 5th byte that represents file system type is 07, which is used to represent both NTFS and exFAT file systems. The starting sector of the file system (where the VBR is located) is the same as before. When converted, it is 63. Let us jump to sector 63.
8. The table below provides the location of the artifacts based on their offset values for the entire volume as well as for the sector. During this exercise, we will use data from the Dec column to locate the relevant entries in sector 63.

Offset		**Universal** Offset		**Length (Bytes)**	**Name**	**Description**
Hex	**Dec**					
0x7E00	32256	0x00	0	3	Jump Instruction	Jump instructions to skip to boot code field
0x7E03	32259	0x03	3	8	OEM ID	ASCII - exFAT
0x7E0B	32267	0x0B	11	53	Must be zero	Replace FAT BIOS parameter block
0x7E40	32320	0x40	64	8	Partition Offset	Sectors from the start of the media
0x7E48	32328	0x48	72	8	Volume Length	Total sector in the volume
0x7E6C	32364	0x6C	108	1	Bytes per sector	2^N N=Value for Bytes
0x7E6D	32365	0x0D	709	1	Sectors per Cluster	2^N N=Value for sectors
0x7E48	32328	0x64	100	4	Volume Serial Number	Serial number of the Volume
0x7FFE	32766	0x1FE	510	2	Boot Signature	0x55 AA

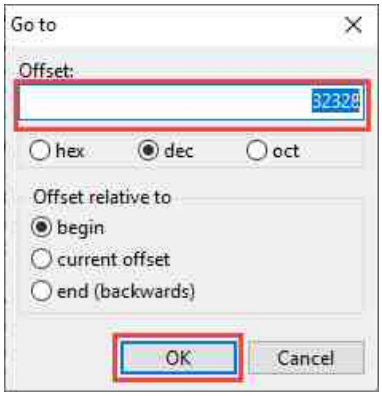
9. Since the calculations for this file system are the same, we already know that the starting sector is located at offset 32256 in our FEF. Let us reopen Go to by clicking the Search dropdown menu option and clicking Go to from the menu or typing Ctrl+G. In the Go to window, type `32256` and click OK as highlighted below.



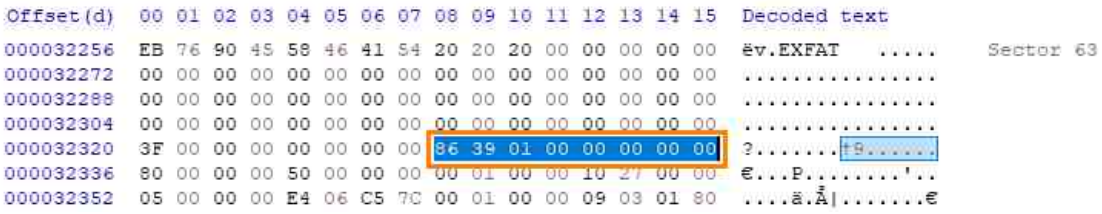
10. At offset 32256 (Sector 63), begin by highlighting the OEM ID. This is located at offset 32259 in our FEF or the 3rd byte from the beginning of the sector. Highlight the following 8 bytes after offset 32259 to reveal the OEM ID as highlighted below. As seen in the textual version of the highlighted data, the OEM ID is EXFAT, which indicates that it is an exFAT file system.

Unlike the other file systems, the next 53 bytes that follow the OEM ID will always be 0x00 in exFAT.

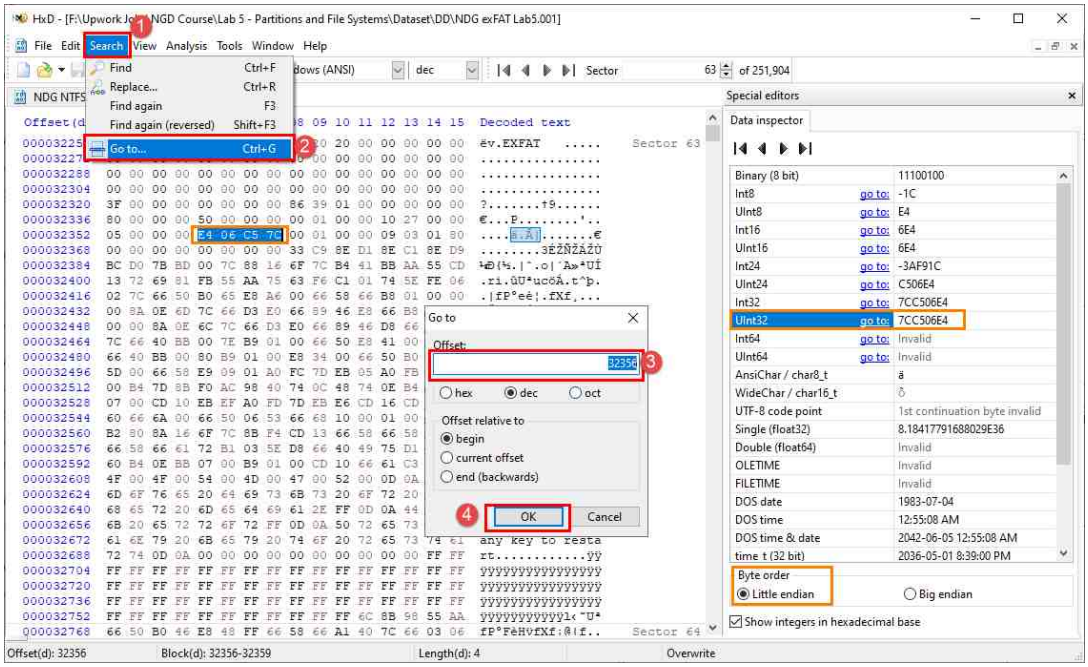
11. Now let us jump to the Volume length artifact. This is located at offset 32328 in our FEF or the 72nd byte from the beginning of the sector. To do this, let us reopen Go to by clicking the Search dropdown menu option, and clicking Go to from the menu or typing Ctrl+G. In the Go to window, type `32328` and click OK as highlighted below.



12. Let us highlight the next 8 bytes, as seen below. This value is represented as 0x86 39 01 00 00 00 00 00, which, when converted to decimal, is 80262. This indicates that there are 80262 sectors. The size in bytes can be found by multiplying the number of sectors (80262) by the sector size (512), which is equal to 41,094,144 bytes (41MB).

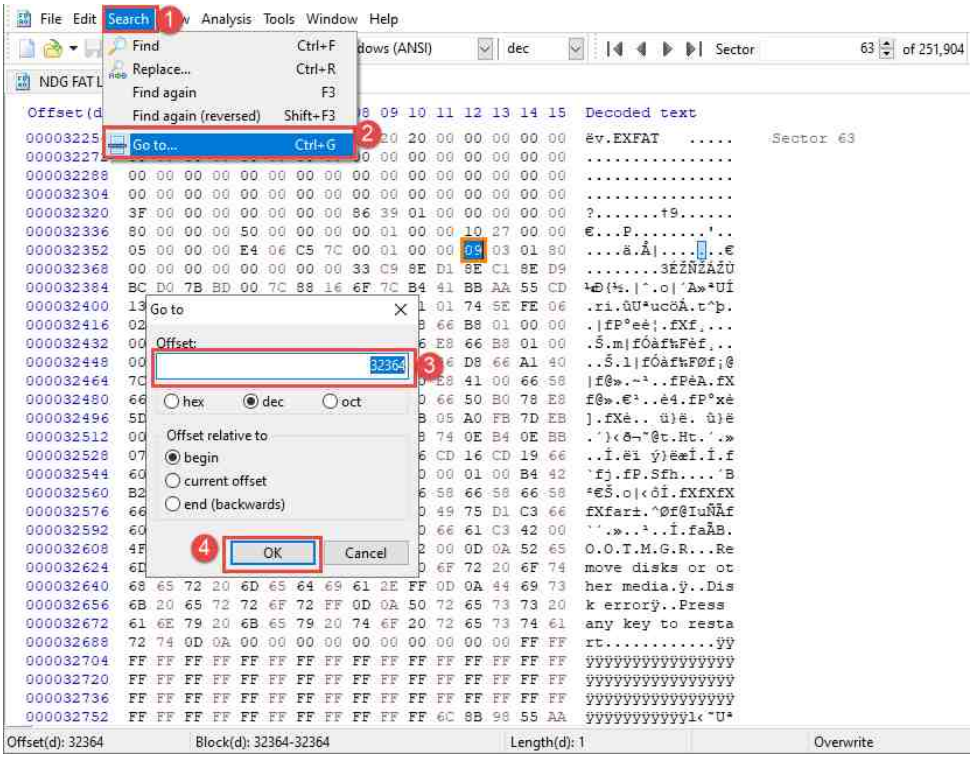


13. Next, let us identify the volume serial number for this file system. This can be found at the 100th byte from the beginning of the sector or offset 32356 in our FEF. Let us use Go to by browsing to the Search dropdown menu on the menu bar and clicking Go to or using Ctrl+G. Type `32356` and click OK. Now that you are at the offset 32356 (100th byte), highlight the next 4 bytes as seen below.



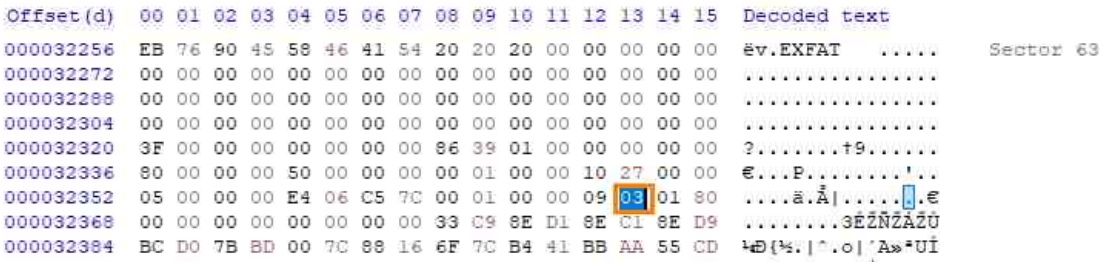
The volume serial number is created when the drive is formatted and can be used to determine if files were ever stored on the drive. These bytes represent the volume serial number and are stored in little-endian. When converted, the volume serial number will read 7CC5-06E4.

14. Now, let us use Go to again to go to the bytes per sector artifact. This can be found at the 108th byte from the beginning of the sector or offset 32364 in our FEF. Reopen Go to by browsing to the Search dropdown menu on the menu bar, and clicking Go to or using Ctrl+G. Type 32364 and click OK. You will be taken to offset 32364. Highlight the byte that follows the cursor as highlighted below.



The exFAT file system stores the sector size as 2 to the power of the value in the VBR. The value highlighted below is 0x09 or 9 in decimal; 2 to the power of 9 is 512 as such the number of bytes per sector on this volume is 512 bytes.

15. Next, let's determine the number of sectors per cluster. This is located at offset 32365 in our FEF or the 109th byte from the beginning of the sector. This means, it is right beside the bytes per sector above. Let us highlight the value. As seen below, the value is 0x03 or 3 in decimal. As with above, the calculation is done by raising 2 to the power of the highlighted value. In this case, the equation is $2^{3,}$ which is equal to 8.



Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Decoded text	
000032256	EB	76	90	45	58	46	41	54	20	20	20	00	00	00	00	00	ëv.EXFAT	Sector 63
000032272	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
000032288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
000032304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
000032320	3F	00	00	00	00	00	00	86	39	01	00	00	00	00	00	00	?.....†9.....	
000032336	80	00	00	00	50	00	00	00	00	01	00	00	10	27	00	00	€...P.....'..	
000032352	05	00	00	00	E4	06	C5	7C	00	01	00	00	09	03	01	80	....ä.Å|.....€	
000032368	00	00	00	00	00	00	00	00	33	C9	8E	D1	8E	C1	8E	D9	.....3ÉŽÑŽÁŽÙ	
000032384	BC	D0	7B	BD	00	7C	88	16	6F	7C	B4	41	BB	AA	55	CD	¼Đ{½.|ˆ.o|´A»ªUÍ	

This means that there are 8 clusters per sector in this volume. Combining this value with the number of bytes per sector can provide the cluster size for the volume. In this case, 512 bytes per sector * 8 sectors = size of each cluster 4096 bytes.

16. As with the previous file system, there are many artifacts that can be unearthed but will not be covered in this exercise.
17. The artifacts you unearthed in this lab can help you get a better understanding of the way data is stored on a drive and determine its structure. Many technical users manipulate the logical volumes to hide data. Mastering this part of the analysis will give you a head start in detecting these attempts and successfully investigating the dataset.

18. Now that you are done, close the Go to window and the HxD program by clicking the X at the top-right corner as highlighted below.