Technical Analysis of Early Personal Computers and its Disk Operating System

Rorabaugh, Tyler L.
CSCI 460, Operating Systems
Montana State University
Fall 2019

Introduction:

This paper will analyze early personal computers (PCs) and its disk operating system (DOS) in an attempt to gain an understanding of how these early PCs worked. This will be done for both historical and analytical purposes. This analysis will be done to aid in the understanding of some of the low level concepts that PCs are built upon. In analyzing these topics, one should also gain an appreciation of both old computer technology, as it was the foundation of modern computer technology, and for new computer technology, as it is a much easier platform to develop programs on.

The focus of the is paper will be early IBM PCs, associated computer clones, and versions of its DOS from the early 1980's. This timeframe is when the PC and DOS gained a lot of popularity and widespread use by consumers businesses. The IBM PC and clones are what is commonly considered a "PC", even though the term "personal computer" can actually refer to other computers such as Apple's line of computers. When the term "PC" is mentioned in this paper, it will be referring to an IBM compatible PC with an x86 processor running a Microsoft operating system like DOS or Windows, rather than any "personal computer".

The Personal Computer:

Early Personal Computers came about in the 1970s and early 1980s. Before personal computers there were large mainframe computers that required special operators to run. These large computing machines were impractical for personal use due to their large size, high cost, and complex operation. As computer hardware technology became smaller, particularly as powerful processors became smaller and less expensive, computers also became smaller and more affordable. Eventually, the size and the cost was no longer a limitation for owning a personal computer and small computers that could fit on desks were eventually developed and sold.

However, personal computers were still complex machines. The average person would not have been able to operate one. This was mostly due to a lack of useful software. In their early days, personal computers were often sold with minimal or no software included. At that time, if a person wanted to make use of a personal computer, he/she would often need to write their own programs for their particular computer model.

As time went on, basic operating systems emerged to run on the popular models of computers. These operating systems provided simple but basic functionality of the computer, including disk access and IO support. This made computers much more useful to their users because they allowed programs and data to be more easily shared across different machines via floppy disks or tapes.

In the early days of personal computers there were many different designs, manufacturers, and operating systems to choose from. While this could be seen as a benefit to the consumer by giving them many options to choose from, it was actually a drawback because there were not common standards from one PC to the next. Adapters were different, hardware was different, software was different and operating systems were different. This meant that if you bought a particular computer, you were stuck with buying parts and software that only worked with that particular model. This meant that there were very few parts to pick from and that the prices of these parts were rather expensive. This all changed with the IBM PC.

The first generation of the IBM PCs emerged in 1981. At first, the IBM PC was nothing special compared to its competitors. However, IBM chose to use open standards with its new PC. This allowed and welcomed other companies to build hardware and software that was compatible with the IBM PC. The DOS operating system, which was already familiar to users of other computers running CP/M, was provided by Microsoft. The combination of an open standard and familiarity with the operating system produced a large number of hardware devices and software packages for the IBM PC. Soon this standard became very important to users because their favorite programs would often only work with an "IBM compatible PC." This is what started the PC movement we know of today.

Shortly after, IBM released the IBM PC XT in 1983. This was essentially the same as the IBM PC, including minor revisions, but, unlike the original PC, it included a built in hard drive. The third generation of the IBM PC came in 1984 with the IBM PC AT. The AT model contained a new processor and chipset, and upgraded drives. The fourth generation PC, the IBM Personal System/2 (PS/2), launched in 1987.

Among the original IBM PC, XT, and AT, there were many clones introduced as "IBM PC compatible." Because IBM was open with its designs of the PC, other PCs emerged to compete with it. Although this may have hurt IBM in the marketplace, it established their design of the PC as a standard. The IBM PC, along with Microsoft's operating systems and Intel's x86 processors made the PC a standard in the 1980s and 1990s and established what is now known as a "PC".

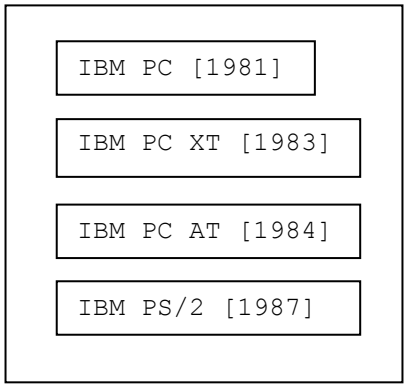


Figure 1: *Timeline of Early IBM PCs*

PC hardware:

Early PCs were much less powerful when compared to modern models. Their hardware was significantly larger and slower. However, they still contained the same basic components that are still used to this day including a processor, memory, drives, and IO such as a keyboard and screen.

Processors:

The advent of powerful new microprocessors in the 1970s and 1980s are what made PCs possible. The processor was the most important piece of the PC. Before microprocessors, computers used large vacuum tubes to perform arithmetic and logic functions. Because a processor requires thousands of these logic units to be useful, early computers took up entire rooms of space. However, with the development of transistors to replace vacuum tubes and then integrated circuits, thousands of these logic units could be stored in a much smaller space. This is what made microprocessors possible.

The first major microprocessor, the Intel 4004, appeared in 1971. This microprocessor is the ancestor of all our modern day x86 processors. The Intel 4004 was a simple 4 bit microprocessor. After the 4004 came the Intel 8008 in 1972, the Intel 8080 in 1974, and the Intel 8085 in 1976. These were 8 bit microprocessors. Then came the 16 bit Intel 8086 in 1978. This chip was the first x86 CPU and is where the x86 architecture originated. Along with the 8086 came the 8088, which was identical to the 8086 except it only had 8 data bus lines instead if 16.

The first important processors of early PCs were these Intel 8086 and the Intel 8088 processors. They were both simple 16 bit processors with 20 bit memory bus lines. The 8086 had a 16 bit data bus line, while the 8088 had an 8 bit data bus line (this means the 8086 could fetch 2 bytes of data at a time instead of 1 byte, like the 8088). Other than this difference, they were essentially the same. The 8088 was the processor inside of the IBM PC, IBM XT, and their clones. The 8087 was an optional floating point processor that could be installed with the IBM PC or IBM XT and clones. The 8087 was designed to work with the 8088 or 8086 to handle floating point operations.

The next important processor in early PCs was the Intel 80286. The 80286 introduced protected mode and memory management capabilities through new circuitry. The address bus was now 24 bits long allowing more memory to be accessed (up to 16MB). Both the data and address busses were also faster than on the 8086 due to extra memory circuits. The 80286 was the processor in the IBM AT computer. The 80287 was an optional floating point processor that could be used to handle floating point operations on IBM AT computers and clones.

Interestingly, many of these advanced features of the 80286 remained unused or unpopular because DOS and its associated software had already gained so much popularity. Many DOS programs were written in a way that made them incompatible with this new protected mode because protected mode did not allow them to access shared memory or hardware directly. It wasn't until the 80386 and early versions of Microsoft Windows that these technologies gained popularity.

Although it is beyond the scope of this paper, the Intel 80386 came out in 1985 and was really the first processor to successfully implement hardware technologies for modern OSs like memory management and multitasking. This processor was a 32 bit processor that included a new virtual 8086 mode to run popular DOS programs that couldn't previously run in the 80286's protected mode. The 80386 also included an integrated memory management unit with paging translation for virtual memory. With the growing popularity of early versions of Microsoft Windows and its programs, these new features became more common. The Intel 80387 was the floating point processor typically used with the 80386.

Beyond these processors are processors like the Intel 80486 and the Intel Pentium, both of which included integrated floating point units and even more technological advancements. However these are outside the scope of this report and were mainly used with the early versions of the Windows operating system.

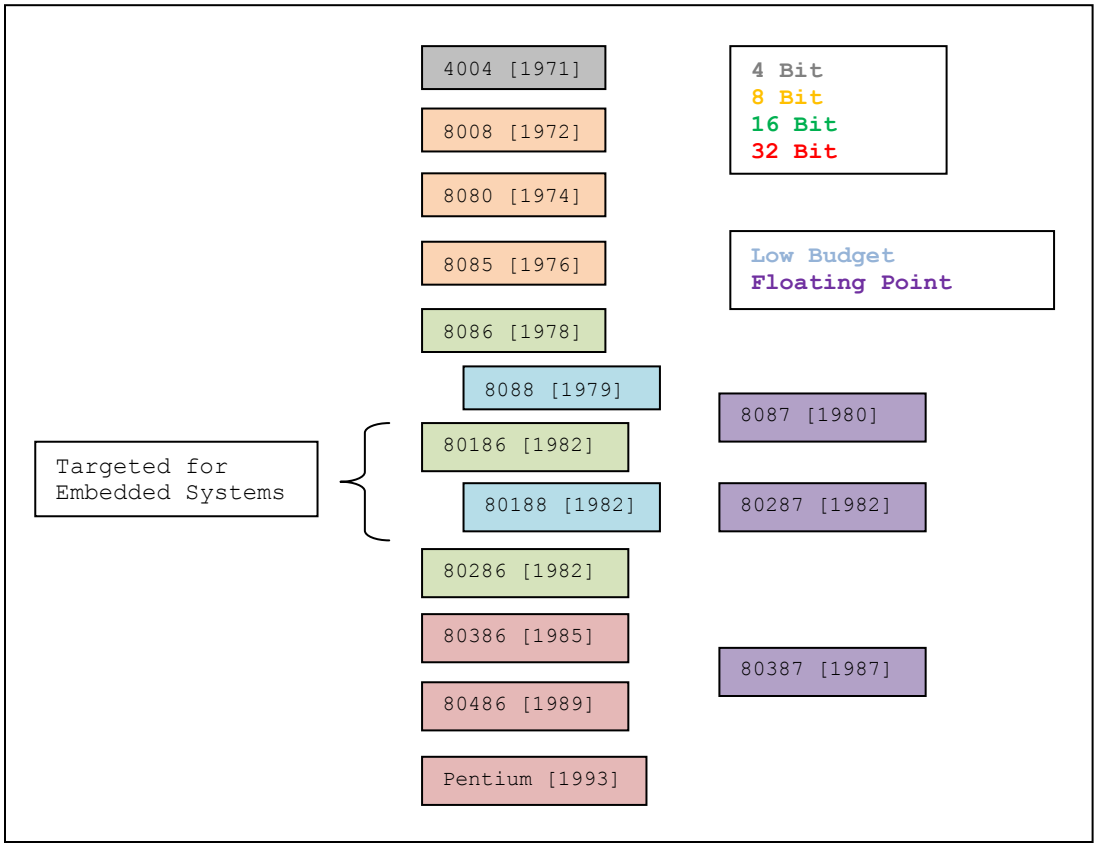


Figure 2: *Timeline of early Intel x86 Processors*

Memory:

The memory configurations of the original IBM PC ranged from 16KB to 256KB of memory. The base model had anywhere from 16KB to 64KB on the motherboard. Three more banks of 64KB could be added on to expansion cards. 8KB of this memory would be used for BIOS ROM code that came with the computer. This left the user with only 8KB to 56KB of memory left.

The IBM XT had slight improvements in memory over the PC. Now the XT could accommodate 1MB of RAM. XTs were first shipped with 64KB of RAM, but as time went on and RAM chips became smaller and cheaper, the XT was packaged with 128KB and eventually 256KB of RAM. Using expansion cards and higher capacity RAM chips also allowed for even more memory, again up to 1MB. Some memory was reserved for use by the BIOS ROM; only 640KB were available to the programmer due to the upper 384KB being used for BIOS functions.

The IBM AT, being newly designed around the Intel 80286, had a larger address 24 bit address bus. Thus the AT could address up to 16MB of memory. Again, this was achievable with higher capacity RAM chips and expansion cards.

Drives:

Due to its low budget nature, the IBM PC was designed to make primary use of tape and floppy drives. Most users used only floppy drives though because they were more convenient than tape drives (tape drives are not random accessible). The original IBM PC did not come with and was not designed for a hard drive. In fact, the original power supply did not allow for it.

One or two floppy drives could be used with the IBM PC and XT. Capacities up to 320KB were possible on 5.25 inch floppy disks Capacities varied weather the disk was single sided or double sided and which version of DOS the PC was running.

The IBM PC originally utilized two 5.25 inch single sided floppy drives. The IBM XT setup was similar to the PC but two 5.25 inch double sided floppies were more common with the XT. However the XT also came standard with a 10MB hard drive.

The IBM AT computer sported higher density 5.25 inch floppy disks. Capacities for these floppy disks could reach 1.2MB. The AT also came standard with a 20MB hard drive.

IO:

The primary input device for all IBM PCs was the keyboard. The keyboard itself was a very important feature of the PC and helped establish it popularity. The keyboards shipped with IBM computers evolved to include a separate number pad, and num-lock and caps-lock buttons with LEDs to identify if they were enabled. A solid, reliable, and easy to use keyboard was important to users of the PC because many were interfacing with their computers for hours on end each day and entering data. A good keyboard made that experience much more enjoyable.

The primary output for the PC was the monitor. The standard monitor for IBM PC was a simple analog color monitor, model 5153. Users could also typically connect to television sets with a special expansion adapter card if they wished to save money. However, the display quality was usually only as good as the video card allowed even if the monitor could support better graphics. Graphics capabilities of the early PCs were usually set by the capabilities of the video card on the system.

The original video cards for the IBM PC were the Color Graphics Adapter (CGA) and Monochrome Display Adapter (MDA). The CGA adapter allowed for 40x25 or 80x25 characters of text display in 16 colors, or 320x200 pixels in 4 colors or 640x200 pixels in 2 colors. The MDA adapter only provided 80x25 characters in 2 colors, with no support for pixel graphics.

In 1985 the Enhanced Graphics Adapter (EGA) was introduced. This adapter was common in the IBM AT. This adapter included new graphics modes, up to 640x350 pixels in 16 colors.

A simple speaker also came standard with the PC. Software could be written to emulate different sounds using the speaker for video games or for enhanced user experience.

Other ports on the PC included two serial ports and a printer port. The serial ports could be used to communicate to other various devices like modems and mice or other specialty hardware for a particular application. The printer port provided was a new design that was controversial at first because it was not standard but it quickly displaced other designs as the PC's popularity grew.

Other devices could be made to interface the PC through expansion slots on the PCs motherboard. As the popularity of IBM computers grew, many other types of inputs were developed for it and interfaced the PC via these slots. Many video games made use of joysticks or game pads. Other specialty equipment was made to interface the PC as well such as cash drawers or special machinery. The expansion slots made the PC a lot more useful to consumers because it allowed their PC to be flexible and adaptable.

Limitations of PC Hardware:

There are inherent limitations to these early PCs due to their hardware. One of the primary limitations was the amount of memory available to address. Due to the Intel 8088 design, a maximum of 1 MB of memory could be addressed on the early PC and XT machines. To make matters worse only 640 KB were available to the programmer due to the upper portions being used for BIOS and video memory. There was of course no virtual memory technology to make up for this lack of memory either. Memory was addressed linearly and directly by the programmer.

Another limitation was storage capacity. Floppy drive capacities did not exceed 1.2 MB. Hard drive capacities did not often exceed 20 MB. Often times, multiple floppy drives had to be inserted at one time to run certain software packages. However as larger capacity hard drives became cheaper this problem went away.

The graphics capabilities of these early PCs were very limited as well. These early PCs could hardly display pictures due to their limited resolutions and color pallets. Later as time went on, new graphics cards and display standards were developed and higher resolution software could be implemented.

There was also no hardware support for multi-tasking or memory protection on the early 8088. With the limited computer technology of the day, the extra cost and overhead of multi-tasking and memory protection was not justified on these early processors. While there was support added with the 80286, it was hardly ever used because it was not backwards compatible. These advanced technologies that we are use to in modern machines we simply not as important 40 years ago. Consumers were happy enough owning a personal computer and running their software at a relatively low cost.

DOS:

DOS stands for disk operating system. It is a simple, single-user, single-task operating system that allows programs to be loaded from disk drives and ran on a computer. The user interface is text based rather than using a graphical interface. It lacks most of the "critical" operating system features discussed in CSCI-460, such as multi-tasking, enhanced memory management, and security, but back in the 1980s, it was enough to get the job done.

DOS has roots in the CP/M operating system. CP/M was a simple monitor operating system for computers in the late 70's. 86-DOS, as it was originally called, was written to run in a similar fashion to CP/M but with faster disk buffer logic and a FAT file system that made it more useful. As implied by its first name, it was targeted to run on the x86 architecture of processors. 86-DOS was originally written by Tim Patterson.

DOS was made popular by the IBM-PC and without the PC, it may have been just another ancient and forgotten OS. When IBM decided to source parts of its new PC design from outside vendors it choose Microsoft to provide the entry level OS. Although there were other operating systems designed for the IBM PC, DOS gained more popularity because it was less expensive and familiar.

Microsoft originally sourced DOS from Tim Patterson. Microsoft made some small changes of their own to DOS and MS-DOS 1.0 was born. For the IBM PC specifically, IBM made a few changes to Microsoft's version and rebranded it to PC-DOS 1.0 and sold it with the IBM PC. MS-DOS continued to be developed by Microsoft for other computer systems as well, with PC-DOS being the specific version used in the IBM PC.

This paper will focus on MS-DOS primarily, as it is the root of the other versions like PC-DOS.

How DOS Works:

Boot Process:

MS-DOS is made up of 3 basic files: IO.SYS, MSDOS.SYS, and COMMAND.COM. IBM renamed two of these files to IBMBIO.SYS and IBMDOS.SYS for their PC version. IO.SYS contains specific information about the computer hardware. MSDOS.SYS is the actual kernel of MS-DOS. COMMAND.COM is the primary shell for MS-DOS that dictates the user interface and command prompt (other shells could be loaded). At the most bare-bones implementation, DOS only needs these 3 files to run.

When a PC is turned on it begins executing code at address 0x0FFFF0. This address points to a jump instruction that contains code to initiate the Power On Self Test (POST) for each component. After the POST is finished the ROM boot loader is loaded into memory and runs. The ROM boot loader begins checking disks for a special boot marker on the first sector of the disk. If a valid marker is found then that sector is loaded into memory and is handed control of the computer. This first sector contains a small amount of code, the disk boot loader, that is responsible for knowing where the operating system is on the disk and loading it into memory.

In the case of DOS, the disk boot loader checks to see if IO.SYS and MSDOS.SYS exist on the first sector of the root directory. If so, the boot loader loads IO.SYS and optionally MSDOS.SYS (implementations differ) into memory. The disk loader then jumps to code in IO.SYS.

IO.SYS is made up of two parts: extra BIOS code and SYSINIT code. The BIOS code contains some basic startup code for the system and a list of basic device drivers to use for things like the console, drives, and clock. The SYSINIT code contains code to check how much memory is available and code to load the MSDOS.SYS kernel into its final place in memory. The BIOS code is ran first and then calls the SYSINIT code to run. After SYSINIT finishes initialization of the system and the kernel, it starts the kernel.

The MSDOS.SYS kernel then sets up the computer for its use. It sets up internal tables and work areas in memory and sets up its interrupt vectors. The kernel also runs initialization functions for each of the drivers in the list provided by IO.SYS. These drivers initialize hardware and external hardware interrupt vectors for use by the kernel or applications. The kernel also check statuses returned by the disk drivers and sets up the necessary disk buffers and data structures. MSDOS.SYS then returns control to SYSINIT.

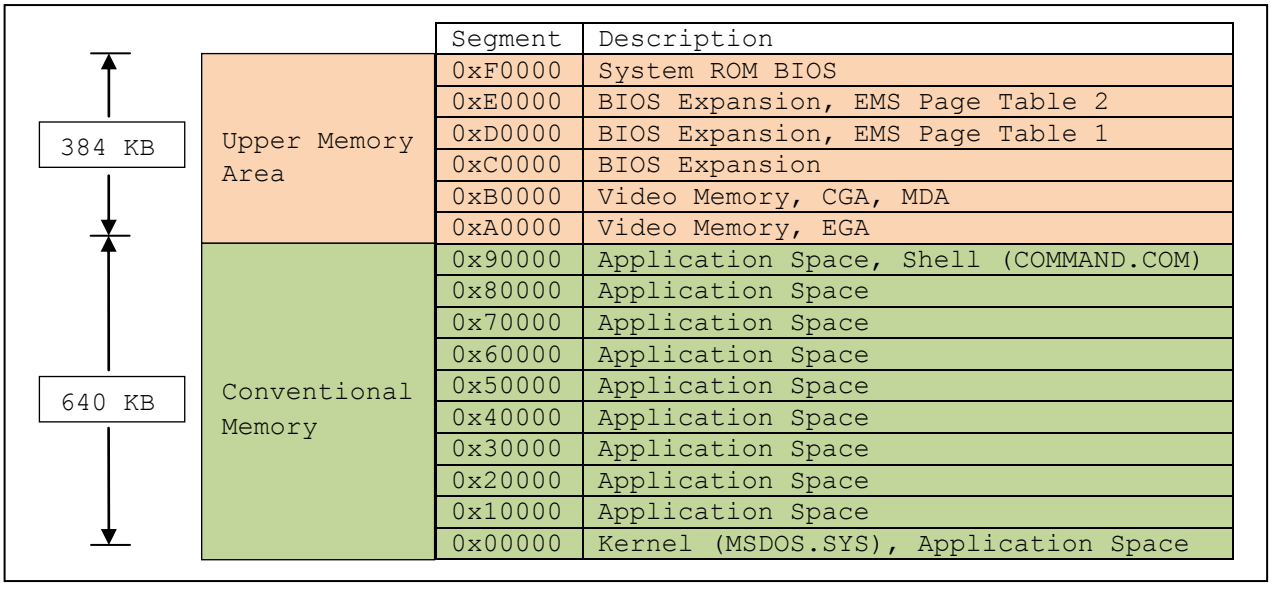
SYSINIT then reads the CONFIG.SYS file (if it is present) with the help of MSDOS.SYS disk reading functions and begins processing the settings specified in CONFIG.SYS. CONFIG.SYS contains extra settings that can be set by the user to customize how DOS will operate. CONFIG.SYS may contain information on how big disk buffers should be, information about file control blocks, and additional device drivers to load and initialize for the system. The CONFIG.SYS file may also contain the name and location of an alternate DOS shell to use (COMMAND.COM is the default).

SYSINIT then closes CONFIG.SYS and opens the console, printer, and auxiliary devices for use as the standard input, standard output, and standard auxiliary devices. Then SYSINIT calls the kernel's execute function to open the shell (COMMAND.COM). COMMAND.COM then displays the "C:>" prompt and waits for user commands.

Memory Management:

DOS memory management is very primitive by today's standards. However it provided a simple way for programs to be loaded and executed on PC's.

A typical memory map of a early PC running DOS looks like the following:



Segment	Description
0xF0000	System ROM BIOS
0xE0000	BIOS Expansion, EMS Page Table 2
0xD0000	BIOS Expansion, EMS Page Table 1
0xC0000	BIOS Expansion
0xB0000	Video Memory, CGA, MDA
0xA0000	Video Memory, EGA
0x90000	Application Space, Shell (COMMAND.COM)
0x80000	Application Space
0x70000	Application Space
0x60000	Application Space
0x50000	Application Space
0x40000	Application Space
0x30000	Application Space
0x20000	Application Space
0x10000	Application Space
0x00000	Kernel (MSDOS.SYS), Application Space

Figure 3: *Typical layout of memory in a PC running DOS*

The original Intel 8088 could address 1MB of memory. On the IBM PC this address space was split into two areas: conventional memory and the upper memory area. Conventional memory is the lower 640KB of address space for general purpose memory that the OS and applications may use. The upper memory area is the upper 384KB of address space that is mapped to ROM BIOS.

At the low end of memory space is where the DOS kernel is located. The interrupt vector table, internal buffers and device drivers are all located here. In the middle of memory, above the location of the DOS kernel, is where the current running application and the shell (COMMAND.COM) are located. The upper portion of memory is reserved for mapping to the BIOS ROM and is used for things like video buffers and other BIOS related data and code. Certain add on cards may use this space to get addressed by the processor as well.

An interesting aside: DOS game and graphics programmers often write screen images directly to the video buffers in this upper portion of memory because it is the fastest way to write pixels to the screen. Although there may have been OS calls to perform these functions for the programmer, they were too slow to be used affectively because of their overhead. As mentioned earlier, this is part of the reason why backwards compatibility with DOS and its applications was so important in the 80386 processor and why the 80286 was not as successful when compared to the 80386. The 80286 did not allow direct access to memory which many programs needed to run properly.

DOS breaks up the free memory into areas called arena entries. Each entry contains a special header to identify its important attributes. When programs are loaded or request additional memory, these arena entries are used. Additionally, all arena entry headers are chained together so that the kernel may quickly search for free entries when allocating space for a program. These arena entries are essentially dynamic partitions similar to the ones we studied in CSCI 460. The default placement algorithm for choosing an arena entry is first fit, but an application can choose other algorithms like best fit or last fit to allocate more memory with later versions of DOS. Arena entries may get filled in different ways when a program is loaded into memory. This allocation is done differently depending on if it is a .COM program or a .EXE program.

.COM programs are raw binary programs that run with minimal overhead. The contain a small header with minimal information and legacy components to be compatible to CP/M (old CP/M programs could be ported to .COM applications with little change). However, they contain no information about segment sizes or memory usage for DOS to use. Therefore these types of programs get allocated the largest entry block of memory at run time (up to 64KB as this is the max amount of memory that a .COM program can effectively use).

A .EXE program does contain information for DOS to use about how much memory it needs and how to segment it. These programs get allocated only the memory they need based on their size and other requests.

Arena entries may also get filled by the allocation of additional memory by DOS programs. DOS provides several functions to applications to allowing them to allocate or de-allocate memory blocks for things like buffers. These functions are accusable by software interrupts to the programmer.

On a typical early IBM PC with an Intel 8088/8086, the amount of usable memory was only 640KB. At first this was plenty of space for programmers to use. But as programs got bigger the 640KB limit became a hindrance. Because DOS and applications shared the conventional memory space, as DOS grew and more OS features were loaded into this space, this left applications decreasing amounts of space to use. Eventually this was partially fixed with by allowing parts of DOS such as the kernel or certain TSRs

(terminate and stay resident programs) to be loaded into portions of the upper memory area. This could be done because some of the spaces in this area were not used anymore and only existed for legacy hardware (like the MDA card). By allowing certain parts of DOS to be "loaded high", this once again gave programmers some more space in conventional memory to use.

However as applications grew and memory became less expensive, conventional memory was still not big enough and continued to be a hindrance on DOS programs. Although outside the scope of this paper, other important memory management features of MSDOS were the additions of Expanded Memory (EMS) and Extended Memory (XMS). These updates allowed more memory to be accessed using special new paging techniques and add on cards. This adaptation by MSDOS to support more memory kept it alive for quite some time, well into the 1990s. Had DOS not been updated, it could have died out much earlier like other operating systems of the day.

File Management:

DOS uses 2 schemes to manage files: File Control Blocks (FCB) and File Allocation Tables (FAT). FCB functionality was inherited from CP/M. This is because DOS was originally designed to be backwards compatible with CP/M's applications and file structures. DOS was designed so that applications that ran on CP/M could be easily ported to run in DOS.

However, FCBs have drawbacks. First, hierarchical file structures were not allowed when using FCBs. There was no way to put groups of file into directories or "folders". When using FCBs in DOS, a program could only access files in its current directory. Because FCBs were just a legacy functionality, using FCBs was not common for applications running under DOS. FAT was superior to the FCB method and was the primary file management system used. Therefore this paper will only explore the details of FAT.

FAT was originally designed in 1977 by Marc McDonald. He used FAT to manage floppy disk files on computers with Intel 8080 processors. When DOS was originally being developed by Tim Paterson, he had knowledge of this FAT file system and implemented it into his DOS.

FAT is a simple and effective way to manage files on drives. FAT uses a index table (the File Allocation Table) to reference file locations on disk. Furthermore, directories are supported using FAT by coding them as another type of file, with references to each of its children files within.

FAT has gone through several iterations, with each standard built on the previous:

FAT	(8 bit)	Very old, not used
FAT12	(12 bit)	Developed for floppy drives, still used some
FAT16	(16 bit)	Developed for larger drives like small hard drives, still used some
FAT32	(32 bit)	Developed for drives up to 4GB, still used for small flash drives and SD cards
exFAT	(64 bit)	Developed for drives larger than 4GB, used for flash drive and SD cards

The number of bits corresponds to how many bits are used in the cluster value to reference cluster locations on disk. In general, a drive implementing a FAT file system will divide the drive into sections shown in Figure 4. These sections will be explained below.

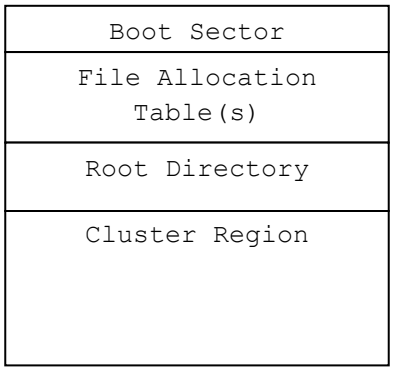


Figure 4: *Drive layout using FAT*

The File Allocation Table exists near the beginning of the drive after the boot sector which exists at the very beginning of the drive, as well as some optional reserved sectors depending on the implementation. Some versions of FAT use 2 copies of the File Allocation Table for backup in case one becomes corrupt. The File Allocation Table itself contains entries for each cluster of the drive. In each entry there is a special cluster value. This cluster value will indicate to the OS weather the cluster is free or is being used by a file.

A cluster is a basic contiguous block of data on a drive. In the early days of drives, clusters were the same size as the sectors on the drive. However, this limited to maximum capacity of the partition, so cluster sizes eventually grew to contain multiple sectors (at the expense of greater internal fragmentation though). All clusters are the same size. The cluster data exists on the drive after the File Allocation Table and Root Directory regions (see Figure 4).

If a cluster value is free (in general indicated by 0x0000, FAT16), the OS can use that cluster area to store a file. Because files may be larger than a cluster size, parts of the file may get assigned to different clusters. The clusters of the same file get chained together in the File Allocation table using pointer to the next cluster entry (see Figure 5). If the cluster value indicates usage by a file (for simplicity 0x0001-0xFFFD, FAT 16) then the cluster value itself is pointer to the next cluster value of the file. In this way file contents can be referenced and chained together by their cluster values. Additionally an end of chain marker (for simplicity 0xFFFF) is used to denote the end of the file chain. Depending on the implementation, some cluster value codes may be reserved to mark bad clusters or other clusters for special use. The OS then uses a simple liner transformation to map the cluster value to a physical location on the drive.

Cluster	Value	Description
0000	0001	File 1, Part 1
0001	0002	File 1, Part 2
0002	FFFF	File 1, End
0003	0005	File 2, Part 1
0004	FFFE	Bad Cluster
0005	0006	File 2, Part 2
0006	FFFF	File 2, End
0008	FFFD	Reserved Cluster
0009	FFFF	File 3, End

Figure 5: *Example of File Allocation Table*

Directories are implemented using Directory Tables. A Directory Table is stored just like a normal file on the drive. However, this file contains a table of information regarding the files that are stored with the directory. Each entry in this table contains the file name, attributes, dates, and its cluster value location. Depending on the implementation other information may be stored in the entry. In older FAT versions these table entries were short and imposed limits on how much information you could store about files (including the 8.3 file name limits).

Because the root directory itself is a directory and is the starting point for all directory searches it is not located in a file, but rather is located at beginning of the drive before the cluster data. That way file searching can start with a reference to the root directory and continue down the directory chains, using the FAT table to reference the physical locations of the files and directories.

FAT has grown from FAT, to FAT12, to FAT16, to FAT32 and most recently to exFAT in order to match the increase in drive capacities. Due to how FAT tables are structured and how big each cluster entry is, each FAT version could only reference so much data. This was partially overcome in the early days by formatting one disk into several logical partitions or drives. This allowed bigger hard drives to be used effectively. With this implementation, each drive still had a boot sector and reserved data regions, but could contain multiple File Allocations Directories, Root Directories, and Cluster Regions. Part of the reserved data area contained this information about drive setups.

IO:

DOS relies a great deal on special BIOS routines and functions for its most basic input and output. Basic disk, keyboard, and screen functions are provided by BIOS. These "functions" can be called from within DOS code (or from program code) using software interrupts. These software interrupts cause the processors to halt program execution, reference the Interrupt Vector Table at the beginning of memory (see Figure 3), and jump to special code sections to do different things (read keys, update monitor).

The BIOS (Basic Input Output System) provides the basic program code to talk to the basic devices needed on a PC. This is very important during the boot process because the OS has no idea what specific hardware it is running on. The BIOS provides basic information of the computer as well as basic usage of input and output device like a screen, keyboard, and drive. Each PC has its own built in BIOS from the factory (located on ROM chips) that allow this basic functionality of the computer.

BIOS code for additional add-on devices of a PC are provided in the form of device drivers. In the old days, this code was shipped with the device on a floppy drive or CD. In modern times drivers can also be found on the internet. The BIOS and device drivers provide a layer of abstraction between the specific hardware of the PC and the operating system and applications.

DOS uses the basic BIOS interface to get access to the screen for displaying text, the keyboard for collecting user input, and the hard drive for retrieving or storing data. DOS uses device drivers to interface additional add-on devices like joysticks, sound cards, and video cards. DOS loads these device drivers using the file CONFIG.SYS in the root directory of the drive. This file, in addition to containing special parameters to initialize the system, has lists of device drivers to load and their locations on the drive. DOS loads these drivers during the boot process (mentioned previously).

By allowing third-party devices to interface DOS and its programs with device drivers, many add-on devices and cards were developed for the IBM PC. This only increased its popularity among users, and is part of the reason why the PC became so popular. With the introduction of the PC, users could now add on special cards to their PC to enhance their experience.

API:

DOS does provide programmers with special functionality of the computer through its Application Programming Interface. This is primarily done through special software interrupts, similar to BIOS function calls, that can be called within a DOS program. Some of the functions provided by DOS include file modifications, writing text to the screen, getting input from the keyboard buffer, printing, program spawning, and memory allocation. This API allowed programmers to focus on solving their individual problems while not worrying about how to interface the computer directly.

However, this functionality was quite limited compared to the rich APIs and libraries we programmers have access to today. Programmers of DOS applications often had to write a lot of their own software to get functionality beyond the basics listed. Additionally, because this was a time before the internet, open source libraries did not exist like they do today. This made programming much more difficult (but arguably more interesting and challenging) than it is today. In response to the limited support and limitations of computer hardware of the day, many programmers developed clever and unique programming hacks to enhance the functionality of their programs and push the limits of the PC hardware.

Limitations of DOS:

Although DOS had many limitations throughout its lifetime, it managed to stay quite popular for a long time because it continuously evolved to include better features. Until Windows took over in the mid 1990s (and even then it still needed DOS to run) DOS was the primary operating system for the personal computer.

No Multi-Tasking:

Because DOS was first designed to load and run single programs on a simple one-processor machine it was never implemented multi-tasking or the ability to run multiple programs at once. The closest implementation DOS got to multi-tasking was its Terminate and Stay Resident (TSR) feature. This feature allowed programs to be saved in RAM for further use by a call to an interrupt. While this isn't true multitasking, it does allow multiple programs to exist in memory for immediate use. Various software modules could take advantage of this feature to give the user the impression that other programs were running in the background.

Only One User:

DOS was never designed to facilitate multiple users. Because it was originally designed as a program loader for a simple computer there was never a need to identify users. As time went on this was a limitation to PCs running DOS because many people often shared a PC and desired a way to separate files and settings. Not having any concept of users was also a security flaw because anyone could start the computer and view or modify files. Eventually this problem was addressed by Microsoft with early versions of Windows but DOS never did implement this functionality.

Security:

Back when DOS was designed, computer security was not nearly as important as it is today. No one in those early days could have justified adding extra hardware and software overhead for security; the processors were simply not fast enough yet. However, as processors got faster and malicious people realized the potential payoffs of breaking into computers, security became more important.

The closest way DOS provides any sort of security is through the file attributes such as System , Hidden and Read-only. System and Hidden attributes protect important files from being seen by the someone who may be a little too curious. Files with these attributes will not be shown to the user by default. Of course these files can be seen with a little extra work and extra command line switches. The Read-only attribute protects important files from being modified. Although again, this could be changed relatively easy by a knowledgeable user. Once again Microsoft responded to this with their development of Windows and its implementations of memory protection and user authentication.

Limited API:

Although DOS provided a basic API to programmers, the API was very primitive. Programmers often had to develop their own custom libraries for each of the programs they made. The result was that many commonly used important functions that did not come with the default API existed in each of the program's code, instead of existing in one place only for all programs to reference and use. The primary example that comes to mind are video and graphics functions. Every single graphics application or video game in DOS had its code written to use the monitor for graphic illustrations. Modern day software now references common graphics libraries to perform this work.

Lasting Impacts of the PC and DOS:

Windows:

I would argue that the largest impacts that early PCs and DOS had on the world were the impacts they had on the Microsoft Windows family of operating systems. The first versions of Windows were actually just graphical shells for the MS-DOS kernel that ran on early PCs.

Until the widespread use of the Windows NT kernel by consumers which was standardized by Windows XP, the consumer versions of Windows, including Windows 3.x and Windows 9x needed MS-DOS to start and run. On these versions of Windows, DOS would load in a similar way as it does on its own, but instead of opening a command terminal, it loaded Windows 3.x or Windows 9x and Windows took over.

Windows was made further popular by the PC itself. Without a good operating system, the fancy PCs and new hardware capabilities that came out in the 1990s would have been useless. The PC needed Windows to effectively take advantage of these new features. The PC, Microsoft's operating systems, and Intel Processors are what made the PC what it is today. Without DOS as its parent and without the demand of PCs, Windows may have never come to existence or became as popular as it is.

FAT:

DOS, and eventually Windows, had a large impact on the popularity of the FAT file system. Because DOS and early versions of Windows used the FAT file system for their drives, FAT has become somewhat of a standard for file storage. Although the modern FAT file systems are not exactly the same as the early versions, as a family of file systems it is one of the most popular and the modern versions owe their existence to the early versions.

Additionally, the PC made the floppy drives as a storage medium very popular. In contrast to floppy drives, hard drives get put in a computer and often never come out. Hard drives and their associated file systems are only relevant to the computer systems they exist on. But a floppy disk can be ejected and its data can be transferred to another system. Therefore its file system needs to be more general purpose and easy to use because floppy disks may be read by many operating different systems. This is the role FAT played. FAT allowed the data generated on a PC to be shared on other systems with other OSes.

Various versions of FAT are used to organize files on SD cards, flash drives, and external hard drives. Camera's commonly use FAT32 or exFAT to store picture data on FLASH memory for other devices to access. The default file system for flash drives is also FAT32 or exFAT. Because FAT is so common, most modern operating systems can read FAT volumes. This make storing files in FAT form one of the best ways to distribute files for use between or on any system.

x86 Architecture:

The x86 architecture powers most of the personal computers in the world today. Processors families like Intel Pentium and Intel Core are processors of the x86 architecture. I would argue that without the popularity of the IBM PC and its clones, and the popularity of DOS and the early versions of Windows, the x86 architecture would not be so popular today.

The IBM PC and it's clones, Microsoft's Operating Systems (DOS and Windows), and the x86 architecture, all grew each other in the 1980s and 1990s. They popularity of each further strengthened the popularity of the others. Because the IBM PC was originally designed to use the first x86 processors and

those PCs grew in wild popularity the x86 architecture naturally became a very popular platform for developers. As new PCs came out and both programmers and users wanted more hardware features and faster processors, Intel kept on developing better and faster processors.

However, backwards compatibility was important for users because they wanted to continue to run their favorite programs on their new PCs. Additionally, it would be a waste of programmer time to have to rewrite their software for every new processor that came out. With this in mind, Intel always designed new processors with backward compatibility. You can still load and run DOS from a CD or flash drive on the latest PCs with no compatibility problems! Microsoft also developed their operating systems to be somewhat backward compatible. Most Windows programs developed on older versions of Windows will still run on modern versions of Windows.

In this way the x86 architecture grew in size and popularity. The popularity of the PC and DOS solidified the x86 presence in the market place among consumers, and backwards compatibility with new features kept the consumers coming back for more.

Bibliography:

Duncan, Ray. *Advanced MSDOS Programming, Second Edition*. Washington: Microsoft Press, 1988. https://www.pcjs.org/pubs/pc/reference/microsoft/mspl13/msdos/advdos/

Meyers, Michael J. *A+ Certification All in One Exam Guide, Second Edition*, pgs. 49-78, 187-226, 457-462 513-564, 657-713, New York: McGraw Hill, 1999.

Wikipedia contributors, "IBM Personal Computer," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=IBM_Personal_Computer&oldid=925257942

Wikipedia contributors, "Influence of the IBM PC on the personal computer market," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Influence_of_the_IBM_PC_on_the_personal_computer_market&oldid=918407516

Wikipedia contributors, "List of Intel microprocessors," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=List_of_Intel_microprocessors&oldid=926811813

Wikipedia contributors, "X86," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=X86&oldid=925812310

Wikipedia contributors, "DOS," *Wikipedia, The Free Encyclopedia* https://en.wikipedia.org/w/index.php?title=DOS&oldid=924819571

Wikipedia contributors, "DOS memory management," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=DOS_memory_management&oldid=912987044

Wikipedia contributors, "Conventional memory," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Conventional_memory&oldid=926239031

Wikipedia contributors, "Design of the FAT file system," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Design_of_the_FAT_file_system&oldid=920681387