

TRAKYA ÜNİVERSİTESİ  
FEN BİLİMLERİ ENSTİTÜSÜ

BINARY SEARCH ALTERNATİFİ ALGORİTMA GELİŞTİRME  
ÜZERİNE BİR ARAŞTIRMA

Gürcan UZAL

DOKTORA TEZİ  
BİLGİSAYAR MÜH. ANABİLİM DALI

Danışman: Prof. Dr. Abdüssamet MARŞOĞLU

1997  
EDİRNE

**TRAKYA ÜNİVERSİTESİ  
FEN BİLİMLERİ ENSTİTÜSÜ**

**BINARY SEARCH ALTERNATİF ALGORİTMA GELİŞTİRME  
ÜZERİNE BİR ARAŞTIRMA**

**Gürcan UZAL**

**DOKTORA TEZİ**

**Danışman  
Prof.Dr.Abdüssamet MARŞOĞLU**

**Bu tez 7./12./1997 tarihinde aşağıdaki jüri tarafından kabul edilmiştir**

**Prof.Dr.Abdüssamet MARŞOĞLU  
(Danışman)**

**Prof.Dr.Mesut RAZBONYALI**

**Yrd.Doç.Dr.Nurşen SUÇSUZ**

## ÖZET

Günümüzde yardımcı bellekler üzerine büyük miktarda bilgi toplamak mümkündür. Bu kadar büyük miktarda depolanmış bilgi içerisinde, istenilen bilgiye kısa sürede erişilerek geri alınması son derece önemlidir.

Bu araştırmada, hızlı arama yaptığı kabul edilen Binary Search algoritmasına alternatif olabilecek yeni bir algoritma geliştirmeye çalışılmıştır. Araştırmamızın esasını bilgisayarlarda ön bellek " cache memory" olarak kullanılan statik RAM'ların yüksek erişim hızından yararlanarak, anahtar dosyadan bir defa okuma yaparak arama süresini azaltmak oluşturmaktadır.

Bu amaçla, ön belleklerin çalışma prensibi, aramada kullanılacak anahtarın adrese çevrilmesi, disk-dosya işlemleri için handle tekniği incelenmiştir.

Ayrıca alternatif algoritmamıza destek vermesi için 64K'lık bir ön bellek arabirim kartı yapılmıştır.

## **SUMMARY**

We can store great many data on the auxiliary memories to day. It is important to get access and retrieve the required information from among the previously stored information in a great quantity.

In this research, I tried to develop a new algorithm that is an alternative to Binary Search algorithm which is considered to search quickly. The principle of our research is based on decreasing the search time reading an access area from the key file making use of the high access speed of the static RAMs which are used as cache memory in computers.

We aimed at studying the working principle of cache memories and converting the key used in searching into address. The handle technique has been studied for disk-file operations.

In addition, to support to our algorithm an interface board by 64K cache memory was constructed.

## TEŞEKKÜR

Çalışmalarım Sırasında, tez danışmanlığımı üstlenip yoğunlaşmış bilimsel enerjisini kullanarak her çeşit yardımda bulunan ve daima destekleyen değerli hocam Prof.Dr.Abdüssamet MARŞOĞLU'na sonsuz saygılarımı sunarım.

Çalışmalarımın her aşamasında daima yardımlarını gördüğüm Yard.Doç.Dr.Aytekin ERDEM'e ve Okutman İbrahim UZUN'a teşekkürlerimi sunarım.

Tezin yazımı sırasında bana yardımlarını esirgemeyen eşim Gülten UZAL'a da ayrıca teşekkür ederim.

## İÇİNDEKİLER

|                                                                     |     |
|---------------------------------------------------------------------|-----|
| ÖZET .....                                                          | I   |
| SUMMARY .....                                                       | II  |
| TEŞEKKÜR .....                                                      | III |
| TABLO DİZİNİ .....                                                  | IV  |
| ŞEKİL DİZİNİ .....                                                  | V   |
| BÖLÜM I. GİRİŞ .....                                                | 1   |
| 1.1. Problem .....                                                  | 1   |
| 1.2. Amaç .....                                                     | 1   |
| 1.3. Önem .....                                                     | 1   |
| 1.4. Varsayımlar .....                                              | 2   |
| 1.5. Sınırlılıklar .....                                            | 2   |
| 1.6. Yöntem .....                                                   | 3   |
| BÖLÜM II. ÖN BELLEK.....                                            | 4   |
| 2.1. Ön Bellek.....                                                 | 4   |
| 2.2. 82385 Yüksek Performanslı 32 Bit Ön Bellek Denetleyicisi ..... | 4   |
| 2.2.1. Doğrudan Haritalanmış Ön Bellek Yapısı veTeknikTerimler...   | 4   |
| 2.2.2. Doğrudan Haritalanmış Ön Belleğin Çalışması .....            | 6   |
| 2.2.2.1. Read Hits (Okuma Yakalamaları) .....                       | 6   |
| 2.2.2.2. Read Misses (Okuma Kaçırmaları) .....                      | 6   |
| 2.3. RAM Ön Belleklemenin Prensibi .....                            | 7   |
| 2.4. Haritalama Metodu .....                                        | 8   |
| 2.4.1. Birleşik Haritalama (Associative Mapping).....               | 8   |
| 2.4.2. Doğrudan Haritalama (Direct Mapping) .....                   | 8   |
| 2.4.3. Grup-Birleşik Haritalama (Set-Associative Mapping) .....     | 9   |
| 2.5. Dinamik Haritalama Algoritmaları .....                         | 9   |

|                                                                                                      |    |
|------------------------------------------------------------------------------------------------------|----|
| 2.6. Ana Belleği Güncelleştirme Algoritması .....                                                    | 10 |
| 2.6.1. Write-Through (Store-Through) .....                                                           | 10 |
| 2.6.2. Write-Back (Copy-Back) .....                                                                  | 10 |
| BÖLÜM III. PC'LER İÇİN 64K'LIK ÖN BELLEK ARABİRİM KARTI .....                                        | 11 |
| 3.1. Devre .....                                                                                     | 11 |
| 3.2. Çalışma Prensipleri .....                                                                       | 12 |
| 3.3. Kart'ın Tasarımında Kullanılan Entegreler ve Görevleri .....                                    | 12 |
| 3.4. Adres Kodunun Çözümü .....                                                                      | 15 |
| 3.5. Ön Bellek Kartına Bilgi Yazılması .....                                                         | 16 |
| 3.6. Ön Bellek Kartından Bilginin Okunması .....                                                     | 17 |
| 3.7. ISA Yolu .....                                                                                  | 17 |
| BÖLÜM IV. ANAHTAR KELİMENİN ADRESE ÇEVİRİLMESİ .....                                                 | 19 |
| 4.1. Anahtar =Adres Adreslemesi .....                                                                | 19 |
| 4.2. Anahtar Çevrimi İçin Algoritma .....                                                            | 19 |
| BÖLÜM V. ÖN BELLEK KARTINA YÜKLENECEK ADRESLERİN SAPTANMASI<br>VE YÜKLENMESİ .....                   | 22 |
| 5.1. Ön Bellek Kartına Yüklenecek Adreslerin Saptanması .....                                        | 22 |
| 5.1.1. Birinci Harflerin Yanına Yazılacak ikinci Harflerin<br>Başlangıç Adreslerinin Bulunması ..... | 22 |
| 5.1.2. İkinci Harflerin Yanına Yazılacak Anahtar Dosya<br>Adreslerinin Bulunması .....               | 22 |
| 5.2. Ön Bellek Kartına Bilgilerin Yükleneceği .....                                                  | 34 |
| BÖLÜM VI. ANAHTAR VE ANA DOSYANIN YARATILMASI .....                                                  | 36 |
| 6.1. Handle Dosya ve Kayıt Organizasyonu .....                                                       | 36 |
| 6.2. Anahtar ve Ana Dosyanın Yaratılması .....                                                       | 39 |
| 6.3. Anahtar ve Ana Dosyaya Kayıt Yapılması .....                                                    | 41 |
| BÖLÜM VII. ANAHTAR DOSYANIN ALTERNATİF VE BINARY SEARCH<br>METODU İLE OKUNMASI .....                 | 44 |
| 7.1. Anahtar Dosyasındaki İlk İki Harfi Aynı Olan Kayıtların Bulunduğu                               |    |

|                                                                        |    |
|------------------------------------------------------------------------|----|
| Bölgenin Alternatif Algoritma ile Bulunması .....                      | 44 |
| 7.2. Anahtar Dosyasındaki İlk İki Harfi Aynı Olan Kayıtların Bulunduğu |    |
| Bölgenin Binary Search Algoritması ile Bulunması .....                 | 61 |
| BÖLÜM VIII. BINARY SEARCH VE ALTERNATİF ALGORİTMANIN                   |    |
| KARŞILAŞTIRILMASI .....                                                | 82 |
| BÖLÜM IX. SONUÇ .....                                                  | 86 |
| KAYNAKLAR .....                                                        | 87 |
| ÖZGEÇMİŞ .....                                                         | 88 |
| EKLER .....                                                            | 89 |



## TABLO DİZİNİ

SIRA NO

SAYFA NO

---

|                                                                                                                                            |    |
|--------------------------------------------------------------------------------------------------------------------------------------------|----|
| Tablo 1. Birinci Harfler ile İkinci Harflerin Başlangıç Adresleri ve İkinci Harflerden A ve B Harflerine Ait Anahtar Dosya Adresleri ..... | 24 |
| Tablo 2. İkinci Harflerden C, Ç ve D Harflerine Ait Anahtar Dosya Adresleri.....                                                           | 25 |
| Tablo 3. İkinci Harflerden E, F ve G Harflerine Ait Anahtar Dosya Adresleri .....                                                          | 26 |
| Tablo 4. İkinci Harflerden G, H ve I Harflerine Ait Anahtar Dosya Adresleri .....                                                          | 27 |
| Tablo 5. İkinci Harflerden İ, J ve K Harflerine Ait Anahtar Dosya Adresleri .....                                                          | 28 |
| Tablo 6. İkinci Harflerden L, M ve N Harflerine Ait Anahtar Dosya Adresleri .....                                                          | 29 |
| Tablo 7. İkinci Harflerden O, Ö ve P Harflerine Ait Anahtar Dosya Adresleri .....                                                          | 30 |
| Tablo 8. İkinci Harflerden R, S ve Ş Harflerine Ait Anahtar Dosya Adresleri .....                                                          | 31 |
| Tablo 9. İkinci Harflerden T, U ve Ü Harflerine Ait Anahtar Dosya Adresleri .....                                                          | 32 |
| Tablo 10. İkinci Harflerden V, Y ve Z Harflerine Ait Anahtar Dosya Adresleri .....                                                         | 33 |
| Tablo 11. Karşılaştırma Sonuçları.....                                                                                                     | 83 |
| Tablo 12. Karşılaştırma Sonuçları .....                                                                                                    | 84 |
| Tablo 13. Karşılaştırma Sonuçları .....                                                                                                    | 85 |

## ŞEKİL DİZİNİ

| SIRA NO                                                                                                     | SAYFA NO |
|-------------------------------------------------------------------------------------------------------------|----------|
| Şekil 2.1. Doğrudan Haritalanmış Ön Bellek Organizasyonu.....                                               | 5        |
| Şekil 2.2. Doğrudan Haritalanmış Organizasyon-386DX Adres Yolu<br>Bit Alanları.....                         | 6        |
| Şekil 3.1. DOS Bellek Organizasyonu .....                                                                   | 11       |
| Şekil 3.2. PC'ler İçin 64K'lık Ön Bellek Kartının Açık Devre Şeması .....                                   | 13       |
| Şekil 3.3. 64K'lık Ön Bellek Arabirim Kartının Baskı Devre Şeması.....                                      | 14       |
| Şekil 3.4. Ön Bellek Entegrelerinin PC'nin Bellek Haritasındaki Yerleşim<br>Adresleri.....                  | 15       |
| Şekil 3.5. ISA Taşıt Yuvası .....                                                                           | 18       |
| Şekil 4.1. Anahtar Kelimenin Adrese Çevrilmesi .....                                                        | 21       |
| Şekil 7.1. Alternatif Algoritma İle İçinde Anahtarın Tamamının Aranılacağı<br>Bölgenin Bulunması .....      | 46       |
| Şekil 7.2. Binary Search Algoritması İle İçinde Anahtarın Tamamının<br>Aranılacağı Bölgenin Bulunması ..... | 62       |

# BÖLÜM I

## GİRİŞ

### 1.1. Problem

Bilgi işlemede en önemli konulardan biri, bilgilerin belli bir düzende saklanması ve bu saklanan bilgilere hızlı bir biçimde ulaşılabilmesidir. Bilgilerin saklanma ve aranması probleminin çözümü için çeşitli yöntemler geliştirilmiştir.

Bilgilerin saklanma şekli (sıralanmış ve sıralanmamış olması) arama sırasında önemli olmaktadır.

### 1.2. Amaç

Amacımız bir defa okuma yaparak, arama süresini azaltacak bir algoritma geliştirmektir.

Bu geliştirilen algoritmaya göre, assembler diliyle yazılacak programın, bilgisayarlarda ön bellek (cache memory) olarak kullanılan, ana belleklerden daha hızlı erişim hızına sahip olan statik RAM'lardan da yararlanarak daha hızlı çalışmasını sağlamaktır.

Bu amaçla cevaplandırılacak sorular şunlardır;

- Ön bellek nedir?
- Ön belleklerden nasıl yararlanabiliriz?
- Arama işleminde on karakterlik bir anahtar kelime kullanılacaktır. Bu yüzden anahtar kelimeyi nasıl adrese çevirebiliriz?
- Anahtarların kaydedilmesi için nasıl bir anahtar dosyası oluşturulacak?
- Anahtar dosyasından aranılacak anahtarın nasıl bulunacağı?

### 1.3. Önem

Önceden depolanmış büyük miktardaki bilgiden, bilgi parçacıkları veya bazı özel kısımları geri alma işlemine arama denir. Normalde biz bilgiyi

aramada kullanılan bir anahtara sahip olan her bir kayıt içinde bölünmüş olarak düşünürüz. Aramanın hedefi verilen arama anahtarına uyum sağlayan anahtarlı bütün kayıtları bulmaktır. Aramanın amacı genellikle işleme için kayıt (sadece anahtar değil) içindeki bilgiye erişmektir.

Aramanın uygulamaları yaygındır ve farklı işlemlerin bir türünü içerir. Örneğin bir banka bütün müşteri hesaplarını izlemeye ve onlarla ilgili çok çeşitli işlemleri kontrol etmek için kayıt ve raporlar arasından müşteriye ait bilgileri aramaya ihtiyaç duyar. Havayolu rezervasyon sistemi de benzer isteklere sahiptir, fakat verinin çoğu oldukça kısa ömürlüdür.

Günümüzde yardımcı bellekler üzerine her geçen gün daha büyük miktarda bilgi depolamak mümkündür. Bu kadar büyük miktarda depolanmış bilgi içerisinden, istenilen bilgi parçacıklarına veya bazı özel kısımlarına çok kısa bir sürede erişilmesi ve geri alınması son derece önemlidir.

Bu araştırma ile günümüzde en hızlı arama yaptığı kabul edilen Binary Search algoritmasından daha hızlı arama yapacak bir algoritmanın geliştirileceği umulmaktadır.

#### **1.4. Varsayımlar**

Bu çalışmada aşağıdaki varsayımlardan (sayıtlardan) hareket edilecektir.

- Alternatif arama algoritmasını geliştirmek için diğer algoritmaların incelenmesi uygun bir başlangıç noktasıdır.
- Devre yapımında kullanılan entegrelerin bacak bağlantıları, çalışması ve teknik özellikleri firma data book'larından anlaşılabilir.

#### **1.5. Sınırlılıklar**

- Turbo assembler programlama dili, aynı konuda yapılan tezler, makaleler, çeşitli kitaplar ve firmaların data book'ları ile sınırlıdır.
- Araştırma, Boğaziçi ve Trakya Üniversitesi kütüphanesi ile sınırlıdır.

## **1.6.Yöntem**

Araştırma literatür tarama modelindedir. Ayrıca varolan durum ve istenilen yeterlilikler arasındaki ilişkiler aranacaktır.



## BÖLÜM II

### ÖN BELLEK

#### 2.1. Ön bellek (Cache Memory)

Günümüzün işlemcileri 25MHz, 33MHz ve daha fazla saat frekanslarında çalışmaktadır. Sonuç olarak dinamik RAM'lerden oluşan PC ana bellekleri CPU'ların bu hızlı erişim sürelerine ayak uyduramamaktadır. Bu saat frekanslarında işlemciler zamanlarının yarısını RAM'i beklemekle geçirmek zorunda olduklarından yetenekleri verimli kullanılmamaktadır.

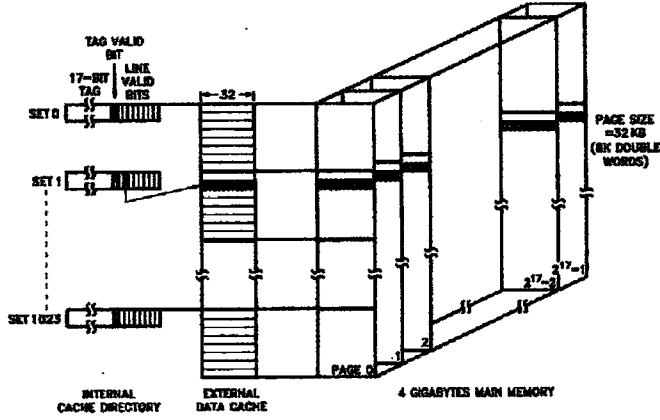
Bu problemi çözmek için bütün ana belleğin daha kısa erişim süreleri olan statik RAM yongalarıyla değiştirilmesi gerekir. Ancak bu çözüm çok pahalıya mal olur. Bunun yerine 1960'larda büyük kapasiteli sistemlerde kullanılan bir yöntem yüksek hızlı 386 ve 486'larda uygulandı. Bu yöntemde bir dış RAM ön bellek tamponu kullanılır. Burada "dış" CPU'nun dışı anlamındadır. Yani ön bellek işlemciye yol sistemi üzerinden bağlanmaktadır. i486 anakartlarında bu dış ön bellek, i486 yongasındaki entegre ön belleğe ek olarak CPU dışında anakart üzerinde bulunduğu için ikinci ön bellek olarak adlandırılır. Bunlar 64 veya 256 kilobit kapasiteli bellek sıralarından oluşur. Bu da ana belleğe göre az miktarda bir bellek olan 64 veya 256K'lık ön bellekler ortaya çıkartır. Bu ön bellek Intel'in 82385 ön bellek denetleyicisi ile yönetilir. [7]

#### 2.2. 82385 Yüksek Performanslı 32 Bit Ön Bellek Denetleyicisi

82385, Basit doğrudan haritalanmış organizasyon "direct mapped organization" ve daha çok az karmaşık yüksek performanslı düzenlenmiş birleşik organizasyon "set associative organization" olmak üzere iki ön bellek organizasyonunu destekler.

##### 2.2.1. Doğrudan Haritalanmış Ön Bellek Yapısı ve Teknik Terimler

82385'in iç ön bellek dizinini, dış ön bellek ve i386DX'in 4 GB fiziksel adres alanı arasındaki ilişki şekil 2.1'de tanımlanmaktadır. 4 GB, herbiri 8 K çiftsözcük (doubleword-32 K bytes) genişliğinde ön bellek sayfaları gibi düşünülebilir. Sayfa büyüklüğü, ön bellek büyüklüğü ile eşleştirilir.

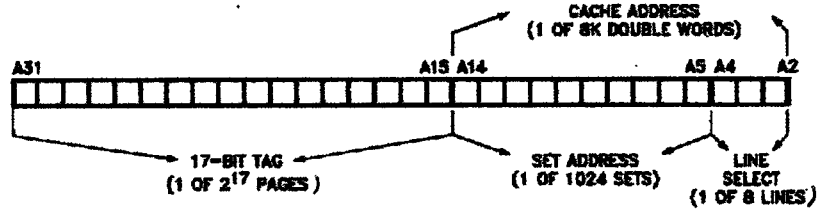


Şekil 2.1. Doğrudan Haritalanmış Ön Bellek Organizasyonu

Ön bellek sekiz çiftsözcük (8x32 bit) olarak düzenlenen 1024 (0 ile 1023) parçaya bölünebilir. Her 32-bit çiftsözcük (doubleword) "line" (hat) olarak adlandırılır. Transferin biri ön belleğin bir hattı ve ana bellek arasındadır. Dış ön bellek içindeki her blok 82385'in iç ön bellek dizini içinde birbiriyle ilişkili 26-bit'lik bir girişe sahiptir. Bu girişler üç kısma ayrılır; 17-bit "tag", 1-bit "tag valid" ve 8-bit "line valid". Tag ana belleğin sayfa numaralamasını yapar (17 tag bit'i  $2^{17}$  sayfayı destekler). Örneğin, eğer ikinci sayfanın 9. hattı halen ön bellek içinde bulunuyorsa, o zaman ikili sistemde bir iki sayısı Set 1'in tag alanı içinde saklanmıştır. (Ana bellek içinde her blok doğrudan haritalanmış ön bellek sayfası için, Set 0 0-7 hatlarından ibarettir, Set 1 8-15 hatlarından ibarettir v.d. 9. hat Şekil 2.1'de koyulaştırılmak sureti ile belirtilmiştir. Bir doğrudan haritalanmış ön belleğin en önemli bir karakteristiği, her sayfanın 9. hattı yalnızca ön belleğin 9. hattı içinde bulunabilir. Bütün özdeş (aynı) sayfa offset'leri tek bir ön bellek mahallini belirtir.

Bir ön bellek içinde düzenlenmiş verinin geçerli veya geçersiz olması, onun "tag valid" bitinin durumuna bağlı olduğu göz önünde bulundurulmalıdır. Eğer temizlenirse (silinirse), bütün düzenleme geçersiz sayılmaktadır. Eğer doğruysa içerisinde düzenlenen yalnızca bir hat, onun "Line valid bits" geçerli hat bitlerinin durumuna bağlı olarak geçerli veya geçersiz olduğu gözönünde bulundurulmaktadır.

82385, i386DX adres yolunu (A2-A31) üç parçaya bölünmüş gibi kabul eder; 17 bit "tag" alanı (A15-A31), 10-bit "set-address" alanı (A5-A14) ve 3-bit "Line select" alanı (A2-A4) (Bakınız Şekil 2.2.).



Şekil 2.2.Doğrudan Haritalandırılmış Organizasyon-386DX Adres Yolu Bit Alanları

Düşük 13 adres biti (A2-A14) aynı zamanda dış ön bellek içinde çiftsözcük 8 K'nın birinin doğrudan seçiminde "cache address" ön bellek adresleri gibi görev yapar.

### 2.2.2. Doğrudan Haritalandırılmış Ön Belleğin Çalışması

386DX, ön bellek ve ön bellek dizisinin birbirine etkisinin açıklanması incelenmektedir.

#### 2.2.2.1. Read Hits (Okuma Yakalamaları)

386DX bir bellek okuma peryodunu başlattığı zaman, 82385 10-bit "Set address" düzenleme adresini 1024 dizinin girişlerinden birini seçmek için ve 3-bit "Line select" hat seçme alanı giriş içerisindeki geçerli hat bitlerinin birini seçmekte kullanılır. 13-bit "cache address" ön bellek adresi ön bellek içinde çiftsözcüğün bulunduğu yeri seçer. 82385 seçilmiş dizinin girişi içinde saklanmış tag ile 17-bit tag alanını (386DX'in erişimindeki A15-A31) karşılaştırılır. Eğer tag ve üst adres bitleri birbirine eşit ise, eğer iki tag ve geçerli hat bitlerini almaya düzenlerse, bir "hit" yakalama meydana gelmektedir ve 82385 ön bellek sürücüsü doğrudan 386DX veri taşıtı üzerindeki çiftsözcüğü seçmektedir. Bir okuma yakalaması dizin ve ön belleğin içeriğini değiştirmez.

#### 2.2.2.2. Read Misses (Okuma Kaçırımları)

Bir okuma kaçırması iki yolla meydana gelebilir. Birincisi bir hat'ı yakalayamaması gibi bilinmektedir ve tag ve üst adres bitleri birbirine eşit ve geçerli tag bit'i düzenlenmektedir, fakat geçerli hat bit'i temizlendiği (silindiği) zaman meydana gelir. İkincisi bir "tag" yakalayamama olarak adlandırılır ve

geçerli tag bit'i temizlendiğinde veya tag ve üst adres bitleri birbirine eşit olmadığı zaman meydana gelir (Geçerli hat bit'i tag kaçırmasında önemli değildir). Her iki durum için, 82385 sistemi 386DX'e bırakır ve veriyi geri getirmeyi i386DX sağlar. i386DX ön bellek dizinini geçerli hale koyar ve ön belleğe yazar.

Bir hat kaçırmasında, ele geçen veri basit olarak önceden geçerli hat bit'nin temizlenmesi (silinmesi) ile geçerli hale konulmaktadır. Bir tag kaçırmasında, üst adres bitleri önceden saklanmış tag üzerine yazılır, geçerli tag biti düzenlenir (ayarlanır), aldığı geçerli hat bitini düzenler ve diğer sekiz geçerli hat biti temizlenir. Sonraki tag yakalamaları ile hat kaçırması yalnızca geçerli hat bit'i tahsisi ile düzenlenecek (Önceden tag ile ilişkilendirilmiş her veri ön bellek içinde uzunca yer alamıyacağı göz önünde bulundurulmalıdır). [1]

### 2.3. RAM Ön Belleklemenin Prensibi

RAM ön belleği CPU ile sistemin ana belleği arasında bulunur ve tampon görevi yapar. IBM uyumlu PC'ler komutları sıralı biçimde yürüttükleri için iyi yazılmış programlar "yerellik prensibine" göre yazılmıştır. Yani program yürütülürken mümkün olduğu kadar birbirine komşu bellek adreslerine erişilir. Daha uzak bellek aralıklarına sıçramalardan kaçınılır.

Birkaç defa yürütülen bir program döngüsünde sürekli aynı komut bellekten okunmak zorundadır. CPU bellekte belli bir yere eriştiğinde bu adres aralığı ve en azından bir komşu adres aralığı daha ön belleğe yazılır ve CPU'ya geçirilir.

Eğer etkili bir programlama tekniği kullanılmışsa CPU'nun bir sonraki bellek erişimi ya aynı bellek aralığına veya komşu bellek aralığına olur. Böylece çoğu zaman gerekli bilgi ön bellekten okunabilir. Aksi halde normal RAM erişimi yapılır.

Bu yöntem CPU'nun ana belleğe erişimini hızlandırır. O anda ön bellekte saklı olan bilgiye erişimi gerektiren işlemlerde hızlı CPU'lar çok daha verimli çalışırlar. Sadece ön bellekte bulunmayan bir bilgi gerektiğinde CPU daha yavaş olan ana bellek kümelerine erişmek zorunda kalır.

Farklı üreticilerin PC anakartları farklı ön bellekleme stratejileri kullanırlar. Bu farklar hangi verilerin saklanacağı ve bulunacağı ve ön bellek bilgilerinin nasıl kullanılacağı gibi konulardadır. Verinin ana bellekten ön belleğe yüklenme biçimi de değişebilir. [7]

## 2.4. Haritalama Metodu

Verinin ana bellekten ön belleğe transferi haritalama işlemi olarak adlandırılır. Haritalama prosedürünün çok önemli üç çeşidi vardır;

- Birleşik haritalama,
- Doğrudan haritalama,
- Grup-Birleşik haritalama

### 2.4.1. Birleşik Haritalama (Associative Mapping)

Bir associative bellek, bellek sözcüğünün içeriğini ve adresin her ikisini de depolar. Bu bellekten herhangi bir sözcüğü (word'ü) ön bellek içinde herhangi bir yere saklamasına imkan verir. Eğer aranan adres ön bellek içinde bulunursa "hit", içerik parçası CPU'ya gönderilir. Eğer bir "miss" (aranılan bilgi ön bellek içinde değilse) oluşursa gerekli adres-veri çifti ana bellekten ön belleğe transfer edilir. Eğer ön bellek doluysa, bir çiftten daha fazla yerleştirilmesine ihtiyaç yoktur. İlk giren ilk çıkar (First-in-first-out = FIFO) gibi, tasarımcılar tarafından kullanılan çeşitli yerine koyma algoritmaları vardır. "Set- associative mapping", ön bellek miss'lerinde harcanan zamanı kısaltır. Çünkü ana bellek adresi ön bellek içinde bilgi ile birlikte depolanır

### 2.4.2. Doğrudan Haritalama (Direct Mapping)

"Associative" bellekler, her bir hücreye göre lojik ilave edildiği için çok pahalıdır. "Random-access" bellekler ayrıca ön bellek gibi kullanılmaktadır.

Adres indeks ve tag olarak iki parçaya bölünmektedir. İndeks alanı içindeki bitlerin sayısı, ön belleğe erişmeye gerekli adres bitlerinin sayısına eşittir. Ön bellek içinde  $2^k$  sözcük ve ana bellek içinde  $2^n$  sözcük vardır. n-bit bellek adresi iki alana bölünmektedir: İndeks alanı k bit ve tag alanı n-k bit, doğrudan haritalama ön bellek organizasyonu ön belleğe erişmeye k-bit indeks ve ana belleğe erişmek için n-bit adres kullanılır. Ön bellek içindeki her sözcük

(word) tag ve data word'u içerir. Bir yeni word ön belleğe getirildiğinde, tag-bit'leri ile data-bit'leri yanyana saklanmıştır. Doğrudan-haritalanmış (direct-mapped) ön bellekler, ön bellek hit'lerinde harcanan zamanı azaltır, çünkü ön bellekte arama için yalnızca bir yer vardır.

### 2.4.3. Grup-Birleşik Haritalama (Set-Associative Mapping)

Grup-Birleşik haritalama ön belleğin herhangi sözcüğünün aynı indeks adresi altında iki veya daha fazla sözcüğü depolayabildiği Doğrudan Haritalama üzerindeki bir iyileştirme. Ön belleğin bir sözcüğünde depolanan "tag-data" parçaları bir grup olarak adlandırılır. Böyle çiftlerin sayısı ön belleğin grup-genişliğidir (Set-size). Hit oranı set-genişliği ile artar, bununla beraber daha fazla karmaşık lojik gereklidir. Seçilmiş "set-associativity" grup genişliği maliyetini minimize etme ve "miss" oranını minimize etme arasında bir uzlaşmadır. Bir küçük grup-genişliği bir grup içindeki pozisyonların küçük sayısı için etkin satırlar arasında yarışma ile ortaya çıkar. Büyük grup genişlikleri arama ve yer değiştirme için daha fazla lojiği gerektirir. "Hit" oranı genişlettilirken grup-genişliğindeki artışlar erişim zamanını arttırabilir ve böylece performansı baştan başa azaltabilir. Grup-Birleşik haritalama ismi aynı indeks'e sahip olan sözcüklerle tag karşılaştırmasının bir birleşik araştırması tarafından yapılmasından oluşur. İki-yollu ve dört-yollu "Set-associative cache" sıkça kullanılır. "Set-associative mapping" küçük ön belleklerde daha iyi performansa sahiptir. Oysa "direct-mapping" daha büyük ön belleklerde daha iyi performans verir.

### 2.5. Dinamik Haritalama Algoritmaları

"Associative", "direct" ve "set-associative mapping" algoritmaları bellek içinde verilen bir hat (line) ön bellek içinde daima aynı grup veya yere yerleştirildiği için statik haritalama algoritmaları olarak adlandırılırlar. Dinamik haritalama algoritmaları bellekten her bir yakalamadan farklı olarak verilen bir satırı rastgele haritalayabilir. Bununla beraber dinamik haritalama, karmaşık haritalama tablolarını gerektirir. Simülasyon sonuçları, dinamik haritalama algoritmalarının statik algoritmalarından az miktarda daha iyi performansa sahip olduklarını gösterir.

## **2.6. Ana Belleği Güncelleştirme Algoritması**

CPU ön bellekteki bir sözcüğü güncelleştireceği zaman iki alışılmış yöntem vardır. "Write-through" (Store-Through) ve "Write-Back (Copy-Back) metodu.

### **2.6.1. Write-Through (Store-Through)**

"Write-Through" yönteminde ön bellek ve ana belleğin her ikisi paralel olarak güncelleştirilirler. Böylece bellek ve ön bellek daima aynı bilgiyi ihtiva eder. Direkt Bellek Erişimli (Direct Memory Access = DMA) sistemler, Giriş/Çıkış aygıtının en son güncelleşmiş bilgiyi verebilmesi için bu karakteristiği gerektirir.

### **2.6.2. Write-Back (Copy-Back)**

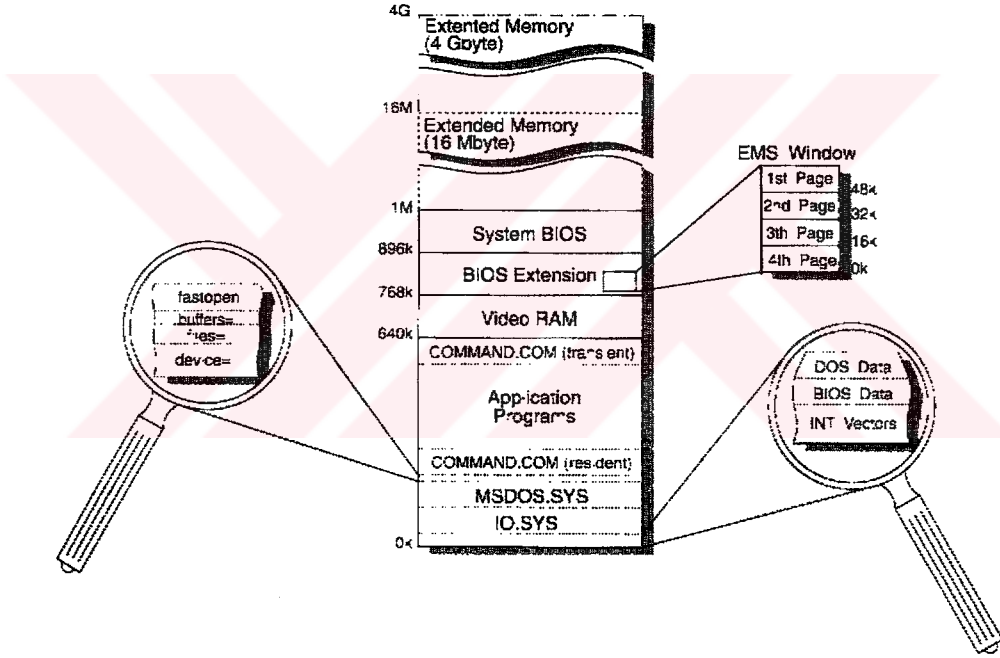
"Write-Back" metodunda yalnızca ön bellek güncelleştirilir ve ön bellek yeri bayraklaştırılır, böylece kelime ön bellekten çıkarıldığında ana belleğe güncelleştirilir. "Write-Back" yöntemi bir sözcük ön bellek içinde olduğu zamanki düşünceden gelir. O birkaç defa güncelleştirilebilir, veri ön bellek içindeyken daha ileri gereksinimler verinin en son biçimi ile ön bellekten karşılanmış olacaktır. Yalnızca kelimeye yer değiştirme yapıldığı zaman bellek güncelleştirilmelidir. "Copy-Back" metodu o ön bellekte yoksa satırın yakalanması için yürütülmelidir. [5]

## BÖLÜM III

### PC'LER İÇİN 64 K'LIK ÖN BELLEK ARABİRİM KARTI

#### 3.1. Devre

DOS bellek organizasyonu incelendiğinde, birinci 640K uygulama programları ve işletim sistemi için ayrılmıştır. Birinci 640K'nın üstünde video RAM vardır ve 768 K'dan başlayan çeşitli BIOS uzantıları izler. 1M'den itibaren uzatılmış bellek başlar, i386, i486 veya Pentium da 4G kadar olabilir.



Şekil 3.1.Dos Bellek Organizasyonu

Video RAM ve genişletilmiş bellek belirtimi (EMS= Expanded Memory Specification) penceresi dışındaki adreslere yalnızca okuma işlemi için erişilebilir. Bellek kartımız için EMS penceresi içinde bulunan D0000-DFFFF adres bölgesini kullanarak, ön bellek kartına istediğimiz bilgileri yükleyebiliriz ve gerektiğinde bu bilgileri buradan alabiliriz. [4]

### 3.2. Çalışma Prensipleri

Mikroişlemci G/Ç arabirimlerinin iki farklı metodu vardır.

- Yalıtılmış G/Ç (Isolated I/O )
- Bellek-Haritalı G/Ç (Memory-mapped I/O)

Yalıtılmış G/Ç'da IN, INS, OUT ve OUTS komutları mikroişlemci akümülatör veya bellek ve G/Ç aygıtları arasında veri transferi yapar.

Bellek-Haritalı G/Ç'da ise IN, INS ve OUTS komutları kullanılmaz. Mikroişlemci ve bellek arasında veri transferi yapan herhangi bir komut kullanılır.

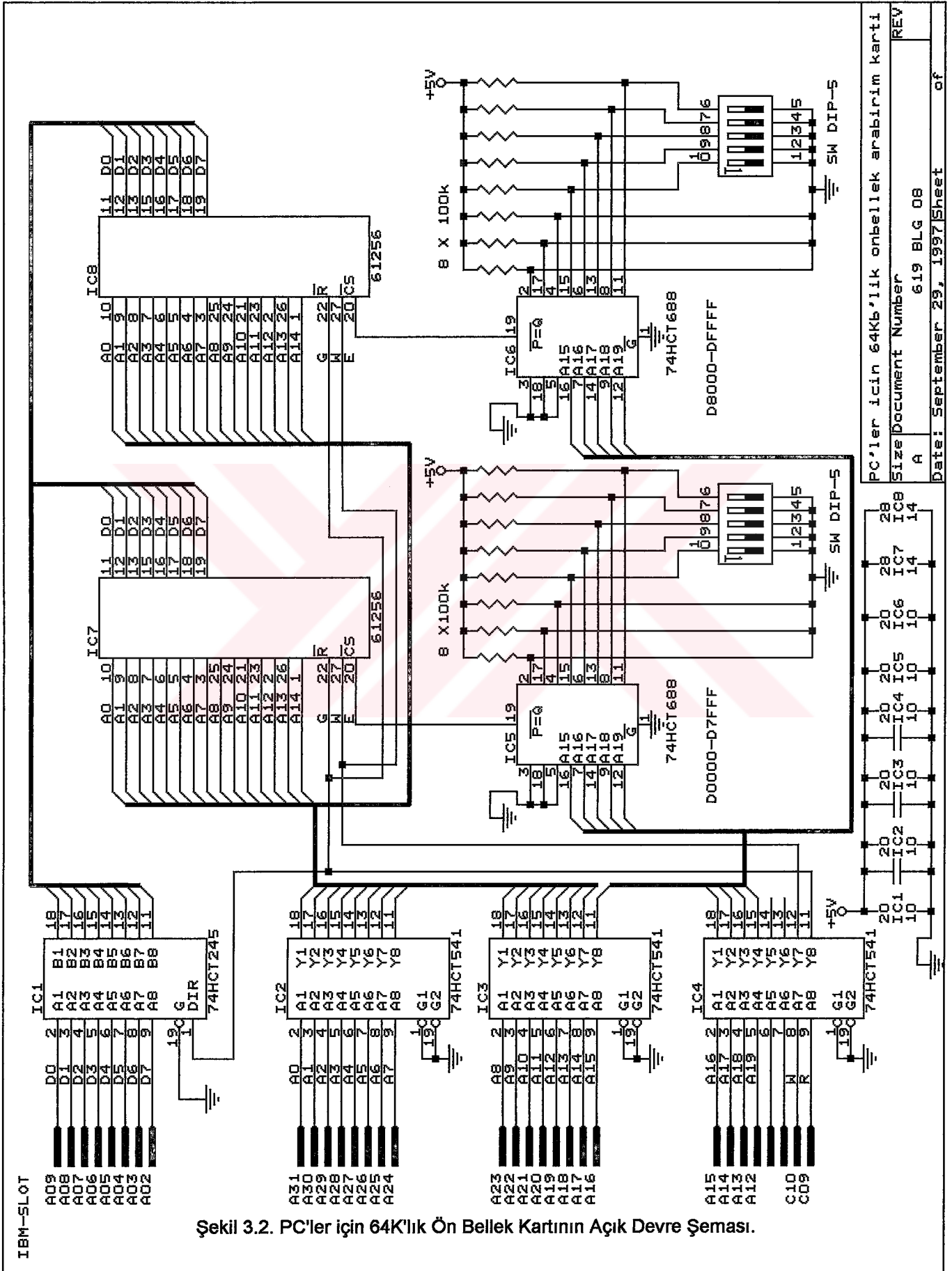
Kartımızın G/Ç portu tasarımında ikinci metod'dan yararlanılmıştır. Çünkü bir bellek haritalı G/Ç portunun, mikroişlemci açısından RAM belleğindeki bir yerden hiç bir farkı yoktur. Bellek-haritalı bir porttan dışarı veri yazılacağı zaman, mikroişlemci sadece port adresine sanki bir RAM yeri adresi gibi veriyi yazar (gönderir). Bellek-Haritalı bir port'tan bilgi içeriği alınacağı zaman, işlemci bir RAM yerinden veri okuyormuş gibi port adresinden veriyi okur.

Tasarlanan G/Ç portu'na ait devreler ve bu port'u kullanan ön bellek entegreleri Şekil 3.2'de gösterilmektedir.

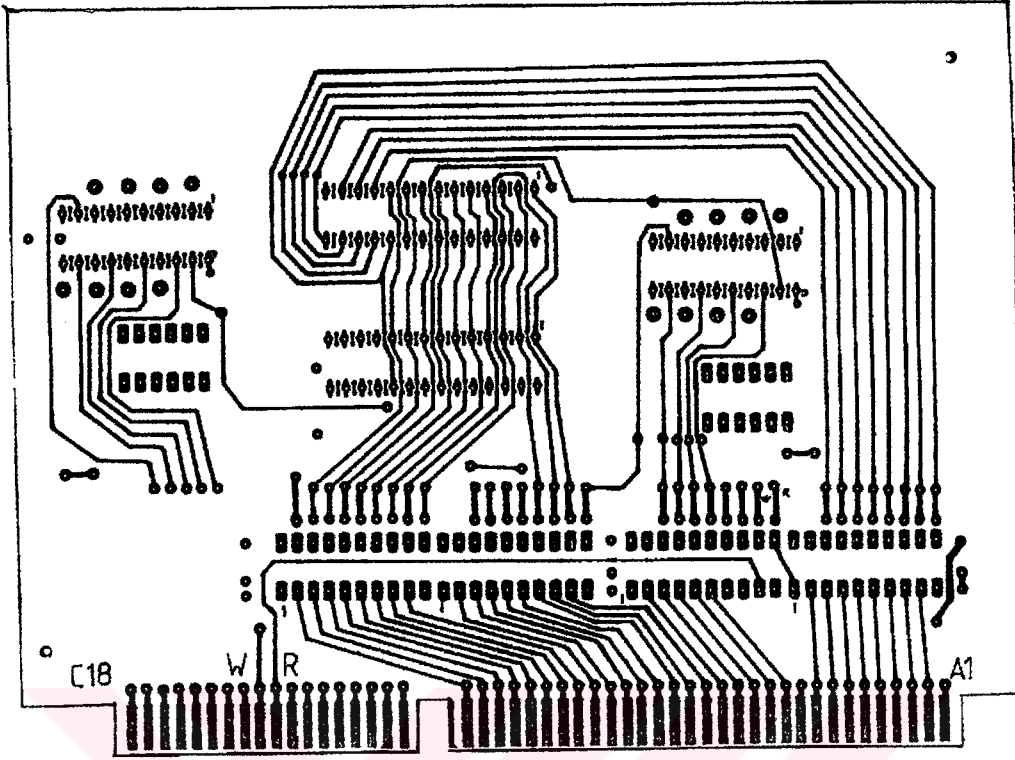
### 3.3. Kartın Tasarımında Kullanılan Entegreler ve Görevleri

**74HCT245:** (Octal bus transceivers with 3-State outputs) üç durumlu sürücüden oluşmuş 8-bit'lik çift yönlü taşıt geçirici entegresi, çift yönlü veri iletimini sağlamak için kullanılmıştır.  $\overline{DIR}$ (direction=yön) bacağı  $\overline{R}$  (read) kontrol sinyaline bağlanmıştır.

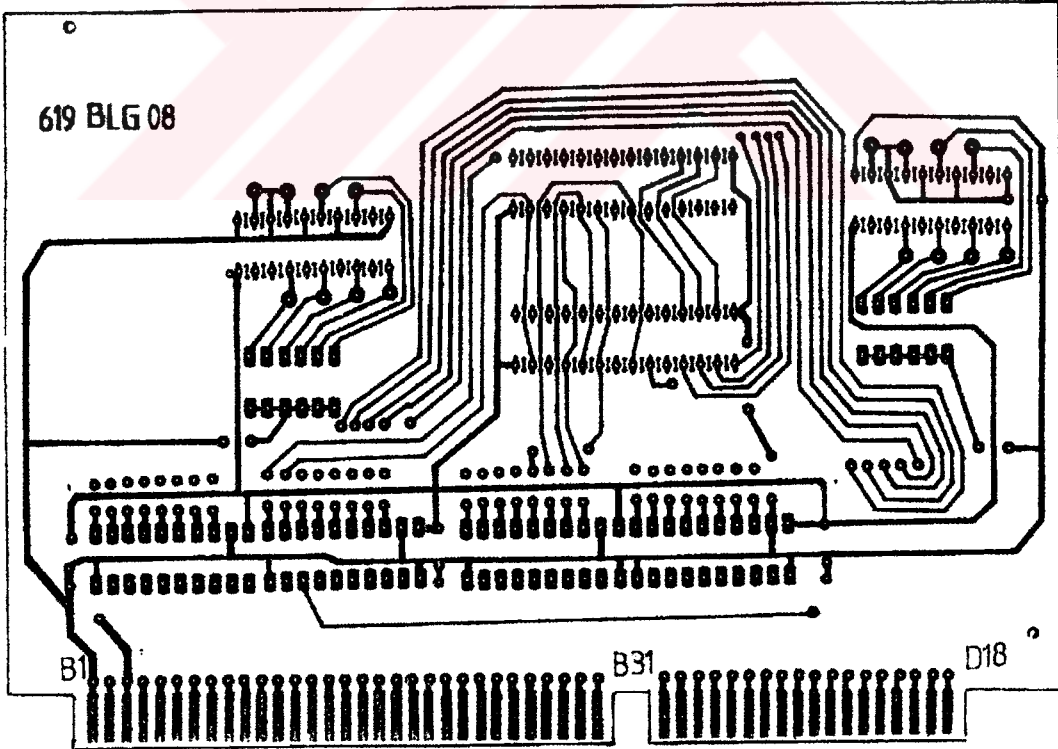
$\overline{R}$  (Read) kontrol sinyali aktif olunca veri taşıtının iletim yönü G/Ç portundan, slot'a doğrudur. Aktif olmadığı zaman ise slot'tan G/Ç portuna doğru olmaktadır.



Şekil 3.2. PC'ler için 64K'lık Ön Bellek Kartının Açık Devre Şeması.



a) Kartın Üst Baskı Devre Şeması



b) Kartın Alt Baskı Devre Şeması

Şekil 3.3. 64 K'lık Ön Bellek Arabirim Kartının Baskı Devre Şeması

**74HCT541:** (Octal buffer and Line driver with 3-State outputs) üç durumlu hat sürücü ve 8 bit'lik tampon entegresi, tek yönlü adres iletimini sağlamak ve adres taşıtını tamponlamak için kullanılmıştır.

**74HTC688:** (8 bit Identity comparators) 8 bit'lik benzerlik karşılaştırıcı entegresi, adres kod çözümünde kullanılmıştır. Dipswitch'le belirlenen adres ile adres taşıtından gelen adres karşılaştırılır. Adresler birbirinin aynı ise entegrenin  $\overline{P=Q}$  bacağı aktif hale getirilir. Bu bacadan alınan sinyal'de SRAM'ı aktif hale getirmek için kullanılmaktadır.

**61256 FK-15:** 32 K-8 bit'lik ve 15 ns'lik ön bellek (SRAM)

### 3.4. Adres Kodunun Çözümü

| Kod çözümü için kullanılıyor |     |     |     |     |     |     |     | Ön belleğin (SRAM'ın) adreslenmesi için kullanılıyor |     |    |    |    |    |    |    |    |    |    |    |  |  |  |  |
|------------------------------|-----|-----|-----|-----|-----|-----|-----|------------------------------------------------------|-----|----|----|----|----|----|----|----|----|----|----|--|--|--|--|
| A19                          | A18 | A17 | A16 | A15 | A14 | A13 | A12 | A11                                                  | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |  |  |  |  |
| 1                            | 1   | 0   | 1   | 0   | 0   | 0   | 0   | 0                                                    | 0   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |  |  |  |  |
|                              | D   |     |     |     | 0   |     |     |                                                      | 0   |    |    |    | 0  |    |    |    | 0  |    |    |  |  |  |  |
| 1                            | 1   | 0   | 1   | 0   | 1   | 1   | 1   | 1                                                    | 1   | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |  |  |  |  |
|                              | D   |     |     |     | 7   |     |     |                                                      | F   |    |    |    | F  |    |    |    | F  |    |    |  |  |  |  |
| 1                            | 1   | 0   | 1   | 1   | 0   | 0   | 0   | 0                                                    | 0   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |  |  |  |  |
|                              | D   |     |     |     | 8   |     |     |                                                      | 0   |    |    |    | 0  |    |    |    | 0  |    |    |  |  |  |  |
| 1                            | 1   | 0   | 1   | 1   | 1   | 1   | 1   | 1                                                    | 1   | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |  |  |  |  |
|                              | D   |     |     |     | F   |     |     |                                                      | F   |    |    |    | F  |    |    |    | F  |    |    |  |  |  |  |

Şekil 3.4.Ön bellek Entegrelerinin PC'nin Bellek haritasındaki yerleşim Adresleri

A0-A14'e kadar olan adres taşıtı ön belleği adreslemek için kullanılmıştır.

A15-A19'a kadar olan adres taşıdı ise SRAM'ların seçilmesi için adres kod çözümünde kullanılmıştır.

74HCT688 entegresi A19-A15 adres hatları ile dipswitch'le belirlenen adresleri karşılaştırır. Dipswitch'le belirlenen adresle, adres taşıtından gelen adresler birbirine eşitse, adres kullanıcı tarafından belirlenen SRAM aktif hale getirilir.

### 3.5. Ön Bellek Kartına Bilgi Yazılması

D000:0000 Adresinden itibaren 1234h, 4567h ve 8910h bilgilerini SRAM'a yüklemek için TASM ile yazılmış bir program örneği aşağıda gösterilmektedir.

```

IDEAL
DOSSEG
MODEL    small
STACK    256
DATASEG
CODESEG

Start:    mov  ax, @data
          mov  ds, ax
          mov  es, ax
          push ds                ; Programın ilk başlatılma
          push es                ; adresinin saklanması
          mov  ax, 0D000h        ; segment adresi
          mov  ds, ax
          mov  es, ax
          mov  di, 0000h        ; offset adresi
          mov  ax, 01234h        ; veri
          stosw                  ; verinin SRAM'a yüklenmesi
          mov  ax, 04567h        ; veri
          stosw                  ; verinin SRAM'a yüklenmesi
          mov  ax, 08910h        ; veri
          stosw                  ; verinin SRAM'a yüklenmesi
          pop  es                ; Programın başlangıç adresinin
          pop  ds                ; yeniden alınması

Exit:     mov  ax, 04C00h        ; Programı bitir
          int  21h              ; DOS ortamına dön
          END  Start

```

### 3.6. Ön Bellek Kartından Bilginin Okunması

D000:0000 adresinden itibaren SRAM'a yüklenen verilerin okunması için TASM ile yazılmış bir program örneği aşağıda gösterilmiştir.

```

        IDEAL
        DOSSEG
        MODEL    small
        STACK    256
        DATASEG
        CODESEG

Start:   mov  ax, @data
        mov  ds, ax
        mov  es, ax
        push ds
        push es
        mov  ax, 0D000h      ;Segment adres
        mov  ds, ax
        mov  es, ax
        mov  si, 0000h      ;offset adresi
        lodsw      ; iki baytlık bilgi akümülatöre yüklenir (ax=1234)
        lodsw      ; iki baytlık bilgi akümülatöre yüklenir (ax=4567)
        lodsw      ; iki baytlık bilgi akümülatöre yüklenir (ax=8910)
        pop  es
        pop  ds

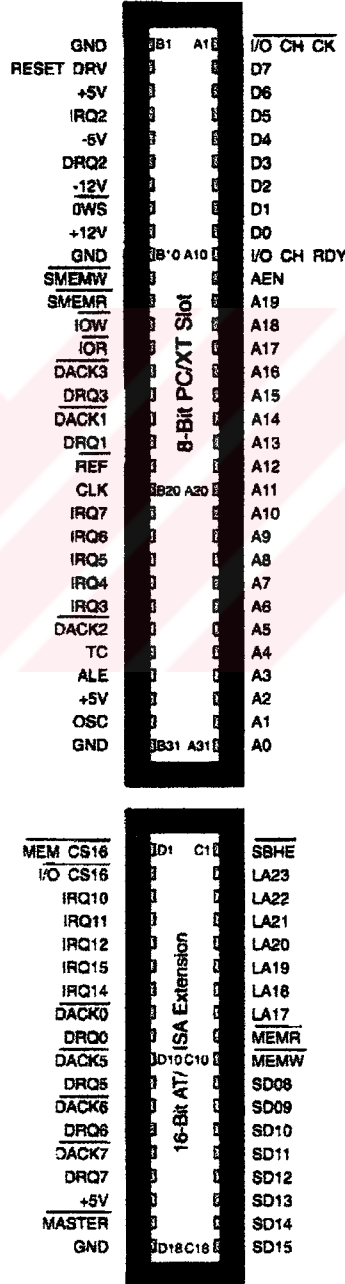
Exit:    mov  ax, 04C00h      ; Programı bitir
        int  21h             ; DOS ortamına dön
        END  Start

```

### 3.7. ISA Yolu

XT Yolu özet olarak 8 bit veri (data) yolu 20 bit adres yolu (1MByte), çeşitli kontrol işaretleri, +12V, -12V, +5V, -5V volt seviyelerinde besleme hatları, saat (Clock) hattı ve toprak hatlarını içeren 62 hatlık (ve dolayısıyla 31 pozisyonlu ve 62 bacaklı konnektör kullanan) yol standardıdır.

ISA (Industry Standard Architecture) Yolu ise XT yolunun genişletilmiş şekli olup 16 bit veri yolu, 24 bit adres yolu (16MByte), ek kontrol işaretleri içerir. 8MHz hızda çalışabilir, iletişim ise 6.5 MByte/saniye'ye çıkabilir. Adaptör kartları; ses, modem v.s. genelde bu iletişim hızlarına göre çok yavaştır. Dolayısıyla pek çok uygulama için bu yol standartı yeterlidir. ISA yolu 98 hatlık (62 bacaklı XT konnektörüne ilaveten 18 pozisyonlu ve 36 bacaklı ikinci bir konnektör kullanılır).



Şekil 3.5. ISA taşıt yuvası.

## BÖLÜM IV

### ANAHTAR KELİMENİN ADRESE ÇEVİRİLMESİ

#### 4.1. Anahtar = Adres Adreslemesi

Bir anahtarı bir dosya içindeki adrese çevirmeye direkt (doğrudan) adresleme denir. Doğrudan adresleme, adreslemeyi en hızlı vasıtaıyla sağlar ve indeks işlemlerini veya dosya araştırmalarını gerektirmez.

Bir anahtarı adrese çevirmeninde çeşitli yöntemleri vardır.

#### 4.2. Anahtar Çevrimi için Algoritma

Bazı uygulamalarda adres, tarih ve havayolu uçuş sayısı veya sayı ve cadde mahalli gibi mevcut bilgilerden hesaplanabilir. Bu, uygulamaların çoğunda mümkün değildir. Fakat uygulanabildiği yerde basit ve hızlı bir yöntemdir. O en çok, dosya referans zamanının kritik olduğu etkileşimli sistemlerde müşterek olarak kullanılır.

Bu yöntem genellikle dosyayı tamamen dolduramama gibi dezavantaja sahiptir. Bir anahtar kelimenin içinde birçok harf vardır. Anahtar kelimenin herbir harfi için 29 değişik durum söz konusudur. Anahtar kelimenin ilk iki harfini sıra numarasına çevirmek istersek, iki harfin  $29 \times 29 = 841$  farklı durumu söz konusudur.

Birinci harfin  $28 \times 29 = 812$  farklı durumu, ikinci harfin 28 farklı durumu ve "1"ilave edildiğinde,  $812 + 28 + 1 = 841$  farklı durum vardır.

$$\text{Sıra no} = \text{Kayıt no} = (L-1) \times 29 + (M-1) + 1$$

Anahtar kelimenin ilk iki harfini sıra numarasına (kayıt numarasına) çevirmek için yukarıdaki formülü kullanabiliriz.

Burada L harfi, anahtar kelimenin birinci harfinin alfabenin kaçınıcı harfi olduğunu, M harfi anahtar kelimenin ikinci harfinin alfabenin kaçınıcı harfi olduğunu belirtmektedir. Alfabenin harflerine birer numara verilir.

(A=1, B=2, C=3, Ç=4, D=5, E=6, F=7, G=8, Ğ=9, H=10, I=11, İ=12, J=13, K=14, L=15, M=16, N=17, O=18, Ö=19, P=20, R=21, S=22, Ş=23, T=24, U=25, Ü=26, V=27, Y=28, Z=29). Anahtar kelimenin birinci harfi, alfabenin harfleri ile karşılaştırılır birbirine eşit olduğunda, L değişkenine harfin sayısal değeri (alfabenin kaçınıcı harfi olduğu) atanır, aynı şekilde kelimenin ikinci harfi de alfabenin harfleri ile karşılaştırılır, karşılaştırma sonucu birbirine eşitse M değişkenine harfin sayısal değeri atanır. L ve M değerleri formülde yerine konulduğunda anahtarın ilk iki harfine ait sıra numarası (kayıt numarası) hesaplanabilir. Şimdi bir kaç örnek verelim;

Örnek;

Anahtar=AAAAAAAAAA ise burada 1. harfin sayısal değeri ve 2. harfin sayısal değeri "1" 'dir. Anahtara karşılık gelen sıra numarası

$$\text{SıraNo} = (L-1) * 29 + (M-1) + 1$$

$$\text{SıraNo} = (1-1) * 29 + (1-1) + 1$$

$$\text{SıraNo} = (0) * 29 + (0) + 1$$

$$\text{SıraNo} = 1$$

Anahtara karşılık gelen sıra numarası AAAAAAAAAA= 1'dir.

Örnek;

Anahtar = ABİ ise burada 1. harfin sayısal değeri "1", ikinci harfin sayısal değeri "2" dir. Anahtarın ilk iki harfine karşılık gelen sıra numarası;

$$\text{SıraNo} = (L-1) * 29 + (M-1) + 1$$

$$\text{SıraNo} = (1-1) * 29 + (2-1) + 1$$

$$\text{SıraNo} = (0) * 29 + (1) + 1$$

$$\text{SıraNo} = 2$$

anahtara karşılık gelen sıra numarası ABİ=2'dir.

Eğer anahtar kelimenin ilk üç harfini sıra numarasına çevirmek istersek, üç harfin maksimum  $29 \times 29 \times 29 = 24.389$  farklı durumda yanyana gelme olasılığı vardır. Bunu şu şekilde formülüze edebiliriz.

$$\text{SıraNo} = (K-1) * 841 + (L-1) * 29 + (M-1) + 1$$

Örnek;

Anahtar=ZZZAAAAAA ise 1., 2., ve 3. harfin sayısal değeri 29'dur.

$$\text{SıraNo} = (29-1)*841 + (29-1)*29 + (29-1) + 1$$

$$\text{SıraNo} = 23548 + 812 + 28 + 1$$

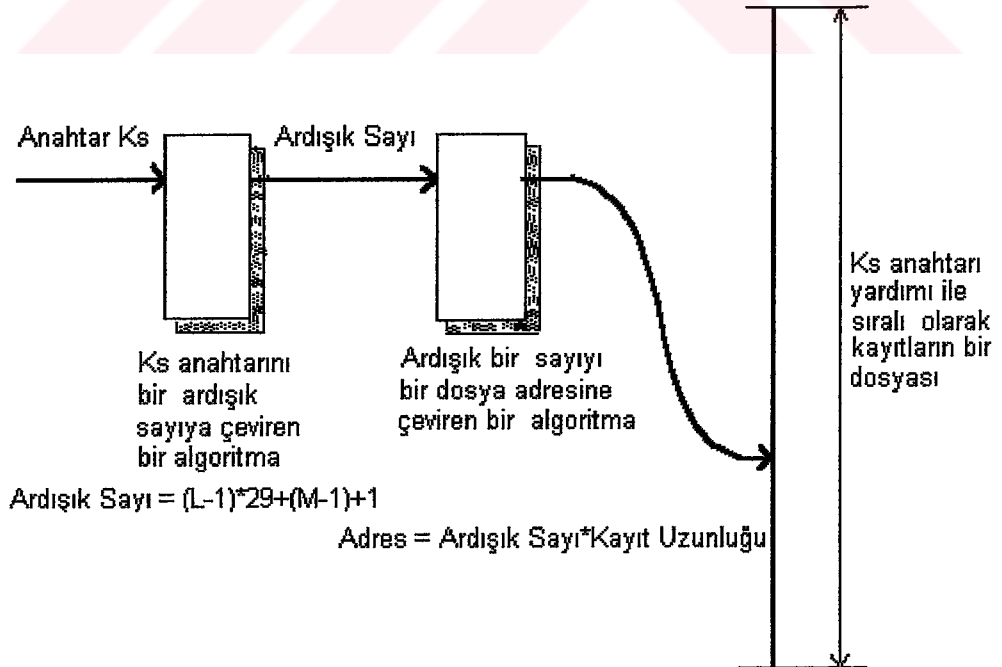
$$\text{SıraNo} = 24389$$

Anahtarı doğrudan adrese çevirmek için sıra numarası ile kayıt uzunluğu çarpılır.

$$\text{Anahtar} = \text{adres} = \text{SıraNo} * \text{Kayıt uzunluğu (bayt olarak)}$$

Bir program bu bağıl adresi bir makina adresine çevirir. Doğrudan adresleme şemalarının bir dezavantajıda onların esnek olmamasıdır. Kayıtların makina adresleri dosyalar genişlediği için değişebilir veya farklı bir birime taşınır veya birleştirilir veya azaltılır.

Yapısal kararlılığa karşı durmak için doğrudan adresleme genellikle iki adımda oluşur. Birincisi anahtarı ardışık bir sayıya çevirir. İkincisi ardışık sayıyı makina adresine çevirir. [3]



Şekil 4.1. Anahtar Kelimenin Adrese Çevrilmesi.

## BÖLÜM V

### ÖN BELLEK KARTINA YÜKLENECEK ADRESLERİN SAPTANMASI VE YÜKLENMESİ

#### 5.1. Ön Bellek Kartına Yüklenecek Adreslerin Saptanması

##### 5.1.1. Birinci Harflerin Yanına Yazılacak İkinci Harflerin Başlangıç Adreslerinin bulunması

Ön bellek kartına D000:0000 adresinden itibaren, bir harf (1bayt) ve ikinci harflerin başlangıç adresleri (2bayt) olmak üzere toplam üç bayt'lık bir bilgi, her harf için yüklenir. Alfabe 29 harf bulunduğundan ve her harf içinde 3 bayt kullanıldığından, toplam olarak ön bellekte ( $29 \times 3 = 87 = 57h$ ) 87 bayt'lık bir yer kaplar. Bu yüzden 0057 offset adresine kadar olan yer, birinci harfler ve ikinci harflerin başlangıç adresleri için kullanılmıştır.

İkinci harflerin başlangıç adresi 0057h'dir. Bu offset adresinden itibaren 1 bayt harf, 4 bayt anahtar dosyasındaki bölge başlangıç adresi ve 2 bayt bölgenin uzunluğu olmak üzere toplam 7 bayt'lık bir bilgi ön belleğe yüklenmektedir. Her ikinci harf ön bellekte ( $29 \times 7 = 203 = CBh$ ) 203 bayt'lık bir yer kaplamaktadır. AA bölgesi 0057h adresinden başlar, eğer 0057h adresini CBh ile toplarsak BA bölgesinin başlangıç adresini ( $0057h + CBh = 122h$ ) buluruz. BA bölgesinin başlangıç adresini yine CBh ile toplarsak, CA'nın başlangıç adresini buluruz. CBh ile toplama işlemine ZA'nın adresini ( $ZA = 168Bh$ ) bulana kadar devam ederiz. Hesapladığımız bu adresleri ilgili birinci harflerin yanına yükleriz.

##### 5.1.2. İkinci Harflerin Yanına Yazılacak Anahtar Dosya Adreslerinin Bulunması

Anahtar dosya adresleri, anahtar kelimenin ilk iki harfine göre belirlenmektedir. İki harfin yanyana değişik durumlarda bulunma olasılığı  $29 \times 29 = 841$ 'dir. Bu nedenle anahtar dosya birbirine eşit 841 bölgeye bölünmektedir. Bu bölgelerin uzunluğunu kullanıcı istediği gibi belirleyebilir. Biz her bölgeyi 141 bayt (8Dh) uzunluğunda aldık. Ayrıca iki harften 1'den 841'e kadar kayıt numarası elde edilmektedir.

Anahtarın ilk iki harfine karşılık gelen kayıt numaralarını kısaca şöyle belirtebiliriz.

AA=001h, AB=002h, AC=003h, AÇ=004h, AD=005h,.....AZ=01Dh  
 BA=01Eh, BB=01Fh, BC=020h, BÇ=021h, BD=022h,.....BZ=03Ah  
 CA=03Bh, CB=03Ch, CC=03Dh, CÇ=03Eh, CD=03Fh,.....CZ=057h  
 . . . . .  
 . . . . .  
 . . . . .  
 ZA=32Dh, ZB=32Eh, ZC=32Fh, ZÇ=330h, ZD=331h,.....ZZ=349h

Anahtar dosyasındaki bölgelerin başlangıç adresleri aşağıda belirtilen şekilde hesaplanır.

$$\text{KayıtNo}=(L-1)*29+(M-1)+1$$

$$\text{Adres}=\text{KayıtNo}*\text{Kayıt Uzunluğu}$$

Örnek;

AA bölgesinin başlangıç adresi,

$$\text{KayıtNo}=(L-1)*1Dh+(M-1)+1=(1-1)*1D+(1-1)+1=1h$$

$$\text{Adres} = 1h * 8Dh = 8Dh \quad ; \text{Adres } 008Dh \text{ olarak bulunur.}$$

Örnek;

ZA bölgesinin başlangıç adresi,

$$\text{KayıtNo}=(L-1)*1Dh+(M-1)+1=(1Dh-1h)*1Dh+(1h-1h)+1=32Ch+1h=32Dh$$

$$\text{Adres} = 32Dh * 8Dh = 1BFC9h \quad ; \text{Adres } 1BFC9h \text{ olarak bulunur.}$$

Adresler ya kayıt numaraları ile kayıt uzunluğunun çarpımı şeklinde yada, formülle birinci bölgenin başlangıç adresi (AA=008Dh) bulunur. Her bölgenin uzunluğu 8Dh olduğu için, birinci bölgenin başlangıç adresi ile toplanırsa AB bölgesinin başlangıç adresi elde edilir. Bu adres yine 8Dh ile toplanırsa AC bölgesinin başlangıç adresi elde edilir. Bu şekilde 8Dh ile toplama işlemine arka arkaya devam edilerek, bütün bölgelerin başlangıç adresleri bulunur.

Ön belleğe yüklenen bütün bilgiler, 1 ve 10 arasındaki tablolarda gösterilmektedir.

| ADRES | BİLGİ  | ADRES | BİLGİ            | ADRES | BİLGİ            |
|-------|--------|-------|------------------|-------|------------------|
|       |        |       | <b>A</b>         |       | <b>B</b>         |
| 0000  | A 0057 | 0057  | A 0000 008D 008D | 0122  | A 0000 1086 008D |
| 0003  | B 0122 |       | B 0000 011A 008D |       | B 0000 1113 008D |
| 0006  | C 01ED |       | C 0000 01A7 008D |       | C 0000 11A0 008D |
| 0009  | Ç 02B8 |       | Ç 0000 0234 008D |       | Ç 0000 122D 008D |
| 000C  | D 0383 |       | D 0000 02C1 008D |       | D 0000 12BA 008D |
| 000F  | E 044E |       | E 0000 034E 008D |       | E 0000 1347 008D |
| 0012  | F 0519 |       | F 0000 03DB 008D |       | F 0000 13D4 008D |
| 0015  | G 05E4 |       | G 0000 0468 008D |       | G 0000 1461 008D |
| 0018  | Ğ 06AF |       | Ğ 0000 04F5 008D |       | Ğ 0000 14EE 008D |
| 001B  | H 077A |       | H 0000 0582 008D |       | H 0000 157B 008D |
| 001E  | I 0845 |       | I 0000 060F 008D |       | I 0000 1608 008D |
| 0021  | İ 0910 |       | İ 0000 069C 008D |       | İ 0000 1695 008D |
| 0024  | J 09DB |       | J 0000 0729 008D |       | J 0000 1722 008D |
| 0027  | K 0AA6 |       | K 0000 07B6 008D |       | K 0000 17AF 008D |
| 002A  | L 0B71 |       | L 0000 0843 008D |       | L 0000 183C 008D |
| 002D  | M 0C3C |       | M 0000 08D0 008D |       | M 0000 18C9 008D |
| 0030  | N 0D07 |       | N 0000 095D 008D |       | N 0000 1956 008D |
| 0033  | O 0DD2 |       | O 0000 09EA 008D |       | O 0000 19E3 008D |
| 0036  | Ö 0E9D |       | Ö 0000 0A77 008D |       | Ö 0000 1A70 008D |
| 0039  | P 0F68 |       | P 0000 0B04 008D |       | P 0000 1AFD 008D |
| 003C  | R 1033 |       | R 0000 0B91 008D |       | R 0000 1B8A 008D |
| 003F  | S 10FE |       | S 0000 0C1E 008D |       | S 0000 1C17 008D |
| 0042  | Ş 11C9 |       | Ş 0000 0CAB 008D |       | Ş 0000 1CA4 008D |
| 0045  | T 1294 |       | T 0000 0D38 008D |       | T 0000 1D31 008D |
| 0048  | U 135F |       | U 0000 0DC5 008D |       | U 0000 1DBE 008D |
| 004B  | Ü 142A |       | Ü 0000 0E52 008D |       | Ü 0000 1E4B 008D |
| 004E  | V 14F5 |       | V 0000 0EDF 008D |       | V 0000 1ED8 008D |
| 0051  | Y 15C0 |       | Y 0000 0F6C 008D |       | Y 0000 1F65 008D |
| 0054  | Z 168B |       | Z 0000 0FF9 008D |       | Z 0000 1FF2 008D |

Tablo 1

| ADRES | BİLGİ            | ADRES | BİLGİ            | ADRES | BİLGİ            |
|-------|------------------|-------|------------------|-------|------------------|
|       | <b>C</b>         |       | <b>Ç</b>         |       | <b>D</b>         |
| 01ED  | A 0000 207F 008D | 02B8  | A 0000 3078 008D | 0383  | A 0000 4071 008D |
|       | B 0000 210C 008D |       | B 0000 3105 008D |       | B 0000 40FE 008D |
|       | C 0000 2199 008D |       | C 0000 3192 008D |       | C 0000 418B 008D |
|       | Ç 0000 2226 008D |       | Ç 0000 321F 008D |       | Ç 0000 4218 008D |
|       | D 0000 22B3 008D |       | D 0000 32AC 008D |       | D 0000 42A5 008D |
|       | E 0000 2340 008D |       | E 0000 3339 008D |       | E 0000 4332 008D |
|       | F 0000 23CD 008D |       | F 0000 33C6 008D |       | F 0000 43BF 008D |
|       | G 0000 245A 008D |       | G 0000 3453 008D |       | G 0000 444C 008D |
|       | Ğ 0000 24E7 008D |       | Ğ 0000 34E0 008D |       | Ğ 0000 44D9 008D |
|       | H 0000 2574 008D |       | H 0000 356D 008D |       | H 0000 4566 008D |
|       | I 0000 2601 008D |       | I 0000 35FA 008D |       | I 0000 45F3 008D |
|       | İ 0000 268E 008D |       | İ 0000 3687 008D |       | İ 0000 4680 008D |
|       | J 0000 271B 008D |       | J 0000 3714 008D |       | J 0000 470D 008D |
|       | K 0000 27A8 008D |       | K 0000 37A1 008D |       | K 0000 479A 008D |
|       | L 0000 2835 008D |       | L 0000 382E 008D |       | L 0000 4827 008D |
|       | M 0000 28C2 008D |       | M 0000 38BB 008D |       | M 0000 48B4 008D |
|       | N 0000 294F 008D |       | N 0000 3948 008D |       | N 0000 4941 008D |
|       | O 0000 29DC 008D |       | O 0000 39D5 008D |       | O 0000 49CE 008D |
|       | Ö 0000 2A69 008D |       | Ö 0000 3A62 008D |       | Ö 0000 4A5B 008D |
|       | P 0000 2AF6 008D |       | P 0000 3AEF 008D |       | P 0000 4AE8 008D |
|       | R 0000 2B83 008D |       | R 0000 3B7C 008D |       | R 0000 4B75 008D |
|       | S 0000 2C70 008D |       | S 0000 3C09 008D |       | S 0000 4C02 008D |
|       | Ş 0000 2C9D 008D |       | Ş 0000 3C96 008D |       | Ş 0000 4C8F 008D |
|       | T 0000 2D2A 008D |       | T 0000 3D23 008D |       | T 0000 4D1C 008D |
|       | U 0000 2DB7 008D |       | U 0000 3DB0 008D |       | U 0000 4DA9 008D |
|       | Ü 0000 2E44 008D |       | Ü 0000 3E3D 008D |       | Ü 0000 4E36 008D |
|       | V 0000 2ED1 008D |       | V 0000 3ECA 008D |       | V 0000 4EC3 008D |
|       | Y 0000 2F5E 008D |       | Y 0000 3F57 008D |       | Y 0000 4F50 008D |
|       | Z 0000 2FEB 008D |       | Z 0000 3FE4 008D |       | Z 0000 4FDD 008D |

Tablo 2

| ADRES | BİLGİ            | ADRES | BİLGİ            | ADRES | BİLGİ            |
|-------|------------------|-------|------------------|-------|------------------|
|       | <b>E</b>         |       | <b>F</b>         |       | <b>G</b>         |
| 044E  | A 0000 506A 008D | 0519  | A 0000 6063 008D | 05E4  | A 0000 705C 008D |
|       | B 0000 50F7 008D |       | B 0000 60F0 008D |       | B 0000 70E9 008D |
|       | C 0000 5184 008D |       | C 0000 617D 008D |       | C 0000 7176 008D |
|       | Ç 0000 5211 008D |       | Ç 0000 620A 008D |       | Ç 0000 7203 008D |
|       | D 0000 529E 008D |       | D 0000 6297 008D |       | D 0000 7290 008D |
|       | E 0000 532B 008D |       | E 0000 6324 008D |       | E 0000 731D 008D |
|       | F 0000 53B8 008D |       | F 0000 63B1 008D |       | F 0000 73AA 008D |
|       | G 0000 5445 008D |       | G 0000 643E 008D |       | G 0000 7437 008D |
|       | Ğ 0000 54D2 008D |       | Ğ 0000 64CB 008D |       | Ğ 0000 74C4 008D |
|       | H 0000 555F 008D |       | H 0000 6558 008D |       | H 0000 7551 008D |
|       | I 0000 55EC 008D |       | I 0000 65E5 008D |       | I 0000 75DE 008D |
|       | İ 0000 5679 008D |       | İ 0000 6672 008D |       | İ 0000 766B 008D |
|       | J 0000 5706 008D |       | J 0000 66FF 008D |       | J 0000 76F8 008D |
|       | K 0000 5793 008D |       | K 0000 678C 008D |       | K 0000 7785 008D |
|       | L 0000 5820 008D |       | L 0000 6819 008D |       | L 0000 7812 008D |
|       | M 0000 58AD 008D |       | M 0000 68A6 008D |       | M 0000 789F 008D |
|       | N 0000 593A 008D |       | N 0000 6933 008D |       | N 0000 792C 008D |
|       | O 0000 59C7 008D |       | O 0000 69C0 008D |       | O 0000 79B9 008D |
|       | Ö 0000 5A54 008D |       | Ö 0000 6A4D 008D |       | Ö 0000 7A46 008D |
|       | P 0000 5AE1 008D |       | P 0000 6ADA 008D |       | P 0000 7AD3 008D |
|       | R 0000 5B6E 008D |       | R 0000 6B67 008D |       | R 0000 7B60 008D |
|       | S 0000 5BFB 008D |       | S 0000 6BF4 008D |       | S 0000 7BED 008D |
|       | Ş 0000 5C88 008D |       | Ş 0000 6C81 008D |       | Ş 0000 7C7A 008D |
|       | T 0000 5D15 008D |       | T 0000 6D0E 008D |       | T 0000 7D07 008D |
|       | U 0000 5DA2 008D |       | U 0000 6D9B 008D |       | U 0000 7D94 008D |
|       | Ü 0000 5E2F 008D |       | Ü 0000 6E28 008D |       | Ü 0000 7E21 008D |
|       | V 0000 5EBC 008D |       | V 0000 6EB5 008D |       | V 0000 7EAE 008D |
|       | Y 0000 5F49 008D |       | Y 0000 6F42 008D |       | Y 0000 7F3B 008D |
|       | Z 0000 5FD6 008D |       | Z 0000 6FCF 008D |       | Z 0000 7FC8 008D |

Tablo 3

| ADRES | BİLGİ            | ADRES | BİLGİ            | ADRES | BİLGİ            |
|-------|------------------|-------|------------------|-------|------------------|
|       | Ğ                |       | H                |       | I                |
| 06AF  | A 0000 8055 008D | 077A  | A 0000 904E 008D | 0845  | A 0000 A047 008D |
|       | B 0000 80E2 008D |       | B 0000 90DB 008D |       | B 0000 A0D4 008D |
|       | C 0000 816F 008D |       | C 0000 9168 008D |       | C 0000 A161 008D |
|       | Ç 0000 81FC 008D |       | Ç 0000 91F5 008D |       | Ç 0000 A1EE 008D |
|       | D 0000 8289 008D |       | D 0000 9282 008D |       | D 0000 A27B 008D |
|       | E 0000 8316 008D |       | E 0000 930F 008D |       | E 0000 A308 008D |
|       | F 0000 83A3 008D |       | F 0000 939C 008D |       | F 0000 A395 008D |
|       | G 0000 8430 008D |       | G 0000 9429 008D |       | G 0000 A422 008D |
|       | Ğ 0000 84BD 008D |       | Ğ 0000 94B6 008D |       | Ğ 0000 A4AF 008D |
|       | H 0000 854A 008D |       | H 0000 9543 008D |       | H 0000 A53C 008D |
|       | I 0000 85D7 008D |       | I 0000 95D0 008D |       | I 0000 A5C9 008D |
|       | İ 0000 8664 008D |       | İ 0000 965D 008D |       | İ 0000 A656 008D |
|       | J 0000 86F1 008D |       | J 0000 96EA 008D |       | J 0000 A6E3 008D |
|       | K 0000 877E 008D |       | K 0000 9777 008D |       | K 0000 A770 008D |
|       | L 0000 880B 008D |       | L 0000 9804 008D |       | L 0000 A7FD 008D |
|       | M 0000 8898 008D |       | M 0000 9891 008D |       | M 0000 A88A 008D |
|       | N 0000 8925 008D |       | N 0000 991E 008D |       | N 0000 A917 008D |
|       | O 0000 89B2 008D |       | O 0000 99AB 008D |       | O 0000 A9A4 008D |
|       | Ö 0000 8A3F 008D |       | Ö 0000 9A38 008D |       | Ö 0000 AA31 008D |
|       | P 0000 8ACC 008D |       | P 0000 9AC5 008D |       | P 0000 AABE 008D |
|       | R 0000 8B59 008D |       | R 0000 9B52 008D |       | R 0000 AB4B 008D |
|       | S 0000 8BE6 008D |       | S 0000 9BDF 008D |       | S 0000 ABD8 008D |
|       | Ş 0000 8C73 008D |       | Ş 0000 9C6C 008D |       | Ş 0000 AC65 008D |
|       | T 0000 8D00 008D |       | T 0000 9CF9 008D |       | T 0000 ACF2 008D |
|       | U 0000 8D8D 008D |       | U 0000 9D86 008D |       | U 0000 AD7F 008D |
|       | Ü 0000 8E1A 008D |       | Ü 0000 9E13 008D |       | Ü 0000 AE0C 008D |
|       | V 0000 8EA7 008D |       | V 0000 9EA0 008D |       | V 0000 AE99 008D |
|       | Y 0000 8F34 008D |       | Y 0000 9F2D 008D |       | Y 0000 AF26 008D |
|       | Z 0000 8FC1 008D |       | Z 0000 9FBA 008D |       | Z 0000 AFB3 008D |

Tablo 4

| ADRES | BİLGİ            | ADRES | BİLGİ            | ADRES | BİLGİ            |
|-------|------------------|-------|------------------|-------|------------------|
|       | I                |       | J                |       | K                |
| 0910  | A 0000 B040 008D | 09DB  | A 0000 C039 008D | 0AA6  | A 0000 D032 008D |
|       | B 0000 B0CD 008D |       | B 0000 C0C6 008D |       | B 0000 D0BF 008D |
|       | C 0000 B15A 008D |       | C 0000 C153 008D |       | C 0000 D14C 008D |
|       | Ç 0000 B1E7 008D |       | Ç 0000 C1E0 008D |       | Ç 0000 D1D9 008D |
|       | D 0000 B274 008D |       | D 0000 C26D 008D |       | D 0000 D266 008D |
|       | E 0000 B301 008D |       | E 0000 C2FA 008D |       | E 0000 D2F3 008D |
|       | F 0000 B38E 008D |       | F 0000 C387 008D |       | F 0000 D380 008D |
|       | G 0000 B41B 008D |       | G 0000 C414 008D |       | G 0000 D40D 008D |
|       | Ğ 0000 B4A8 008D |       | Ğ 0000 C4A1 008D |       | Ğ 0000 D49A 008D |
|       | H 0000 B535 008D |       | H 0000 C52E 008D |       | H 0000 D527 008D |
|       | I 0000 B5C2 008D |       | I 0000 C5BB 008D |       | I 0000 D5B4 008D |
|       | İ 0000 B64F 008D |       | İ 0000 C648 008D |       | İ 0000 D641 008D |
|       | J 0000 B6DC 008D |       | J 0000 C6D5 008D |       | J 0000 D6CE 008D |
|       | K 0000 B769 008D |       | K 0000 C762 008D |       | K 0000 D75B 008D |
|       | L 0000 B7F6 008D |       | L 0000 C7EF 008D |       | L 0000 D7E8 008D |
|       | M 0000 B883 008D |       | M 0000 C87C 008D |       | M 0000 D875 008D |
|       | N 0000 B910 008D |       | N 0000 C909 008D |       | N 0000 D902 008D |
|       | O 0000 B99D 008D |       | O 0000 C996 008D |       | O 0000 D98F 008D |
|       | Ö 0000 BA2A 008D |       | Ö 0000 CA23 008D |       | Ö 0000 DA1C 008D |
|       | P 0000 BAB7 008D |       | P 0000 CAB0 008D |       | P 0000 DAA9 008D |
|       | R 0000 BB44 008D |       | R 0000 CB3D 008D |       | R 0000 DB36 008D |
|       | S 0000 BBD1 008D |       | S 0000 CBCA 008D |       | S 0000 DBC3 008D |
|       | Ş 0000 BC5E 008D |       | Ş 0000 CC57 008D |       | Ş 0000 DC50 008D |
|       | T 0000 BCEB 008D |       | T 0000 CCE4 008D |       | T 0000 DCDD 008D |
|       | U 0000 BD78 008D |       | U 0000 CD71 008D |       | U 0000 DD6A 008D |
|       | Ü 0000 BE05 008D |       | Ü 0000 CDFE 008D |       | Ü 0000 DDF7 008D |
|       | V 0000 BE92 008D |       | V 0000 CE8B 008D |       | V 0000 DE84 008D |
|       | Y 0000 BF1F 008D |       | Y 0000 CF18 008D |       | Y 0000 DF11 008D |
|       | Z 0000 BFAC 008D |       | Z 0000 CFA5 008D |       | Z 0000 DF9E 008D |

Tablo 5

| ADRES | BİLGİ            | ADRES | BİLGİ            | ADRES | BİLGİ            |
|-------|------------------|-------|------------------|-------|------------------|
|       | <b>L</b>         |       | <b>M</b>         |       | <b>N</b>         |
| 0B71  | A 0000 E02B 008D | 0C3C  | A 0000 F024 008D | 0D07  | A 0001 001D 008D |
|       | B 0000 E0B8 008D |       | B 0000 F0B1 008D |       | B 0001 00AA 008D |
|       | C 0000 E145 008D |       | C 0000 F13E 008D |       | C 0001 0137 008D |
|       | Ç 0000 E1D2 008D |       | Ç 0000 F1CB 008D |       | Ç 0001 01C4 008D |
|       | D 0000 E25F 008D |       | D 0000 F258 008D |       | D 0001 0251 008D |
|       | E 0000 E2EC 008D |       | E 0000 F2E5 008D |       | E 0001 02DE 008D |
|       | F 0000 E379 008D |       | F 0000 F372 008D |       | F 0001 036B 008D |
|       | G 0000 E406 008D |       | G 0000 F3FF 008D |       | G 0001 03F8 008D |
|       | Ğ 0000 E493 008D |       | Ğ 0000 F48C 008D |       | Ğ 0001 0485 008D |
|       | H 0000 E520 008D |       | H 0000 F519 008D |       | H 0001 0512 008D |
|       | I 0000 E5AD 008D |       | I 0000 F5A6 008D |       | I 0001 059F 008D |
|       | İ 0000 E63A 008D |       | İ 0000 F633 008D |       | İ 0001 062C 008D |
|       | J 0000 E6C7 008D |       | J 0000 F6C0 008D |       | J 0001 06B9 008D |
|       | K 0000 E754 008D |       | K 0000 F74D 008D |       | K 0001 0746 008D |
|       | L 0000 E7E1 008D |       | L 0000 F7DA 008D |       | L 0001 07D3 008D |
|       | M 0000 E86E 008D |       | M 0000 F867 008D |       | M 0001 0860 008D |
|       | N 0000 E8FB 008D |       | N 0000 F8F4 008D |       | N 0001 08ED 008D |
|       | O 0000 E988 008D |       | O 0000 F981 008D |       | O 0001 097A 008D |
|       | Ö 0000 EA15 008D |       | Ö 0000 FA0E 008D |       | Ö 0001 0A07 008D |
|       | P 0000 EAA2 008D |       | P 0000 FA9B 008D |       | P 0001 0A94 008D |
|       | R 0000 EB2F 008D |       | R 0000 FB28 008D |       | R 0001 0B21 008D |
|       | S 0000 EBBC 008D |       | S 0000 FBB5 008D |       | S 0001 0BAE 008D |
|       | Ş 0000 EC49 008D |       | Ş 0000 FC42 008D |       | Ş 0001 0C3B 008D |
|       | T 0000 ECD6 008D |       | T 0000 FCCF 008D |       | T 0001 0CC8 008D |
|       | U 0000 ED63 008D |       | U 0000 FD5C 008D |       | U 0001 0D55 008D |
|       | Ü 0000 EDF0 008D |       | Ü 0000 FDE9 008D |       | Ü 0001 0DE2 008D |
|       | V 0000 EE7D 008D |       | V 0000 FE76 008D |       | V 0001 0E6F 008D |
|       | Y 0000 EF0A 008D |       | Y 0000 FF03 008D |       | Y 0001 0EFC 008D |
|       | Z 0000 EF97 008D |       | Z 0000 FF90 008D |       | Z 0001 0F89 008D |

Tablo 6

| ADRES | BİLGİ            | ADRES | BİLGİ            | ADRES | BİLGİ            |
|-------|------------------|-------|------------------|-------|------------------|
|       | <b>O</b>         |       | <b>Ö</b>         |       | <b>P</b>         |
| 0DD2  | A 0001 1016 008D | 0E9D  | A 0001 200F 008D | 0F68  | A 0001 3008 008D |
|       | B 0001 10A3 008D |       | B 0001 209C 008D |       | B 0001 3095 008D |
|       | C 0001 1130 008D |       | C 0001 2129 008D |       | C 0001 3122 008D |
|       | Ç 0001 11BD 008D |       | Ç 0001 21B6 008D |       | Ç 0001 31AF 008D |
|       | D 0001 124A 008D |       | D 0001 2243 008D |       | D 0001 323C 008D |
|       | E 0001 12D7 008D |       | E 0001 22D0 008D |       | E 0001 32C9 008D |
|       | F 0001 1364 008D |       | F 0001 235D 008D |       | F 0001 3356 008D |
|       | G 0001 13F1 008D |       | G 0001 23EA 008D |       | G 0001 33E3 008D |
|       | Ğ 0001 147E 008D |       | Ğ 0001 2477 008D |       | Ğ 0001 3470 008D |
|       | H 0001 150B 008D |       | H 0001 2504 008D |       | H 0001 34FD 008D |
|       | I 0001 1598 008D |       | I 0001 2591 008D |       | I 0001 358A 008D |
|       | İ 0001 1625 008D |       | İ 0001 261E 008D |       | İ 0001 3617 008D |
|       | J 0001 16B2 008D |       | J 0001 26AB 008D |       | J 0001 36A4 008D |
|       | K 0001 173F 008D |       | K 0001 2738 008D |       | K 0001 3731 008D |
|       | L 0001 17CC 008D |       | L 0001 27C5 008D |       | L 0001 37BE 008D |
|       | M 0001 1859 008D |       | M 0001 2852 008D |       | M 0001 384B 008D |
|       | N 0001 18E6 008D |       | N 0001 28DF 008D |       | N 0001 38D8 008D |
|       | O 0001 1973 008D |       | O 0001 296C 008D |       | O 0001 3965 008D |
|       | Ö 0001 1A00 008D |       | Ö 0001 29F9 008D |       | Ö 0001 39F2 008D |
|       | P 0001 1A8D 008D |       | P 0001 2A86 008D |       | P 0001 3A7F 008D |
|       | R 0001 1B1A 008D |       | R 0001 2B13 008D |       | R 0001 3B0C 008D |
|       | S 0001 1BA7 008D |       | S 0001 2BA0 008D |       | S 0001 3B99 008D |
|       | Ş 0001 1C34 008D |       | Ş 0001 2C2D 008D |       | Ş 0001 3C26 008D |
|       | T 0001 1CC1 008D |       | T 0001 2CBA 008D |       | T 0001 3CB3 008D |
|       | U 0001 1D4E 008D |       | U 0001 2D47 008D |       | U 0001 3D40 008D |
|       | Ü 0001 1DD8 008D |       | Ü 0001 2DD4 008D |       | Ü 0001 3DCD 008D |
|       | V 0001 1E68 008D |       | V 0001 2E61 008D |       | V 0001 3E5A 008D |
|       | Y 0001 1EF5 008D |       | Y 0001 2EEE 008D |       | Y 0001 3EE7 008D |
|       | Z 0001 1F82 008D |       | Z 0001 2F7B 008D |       | Z 0001 3F74 008D |

Tablo 7

| ADRES | BİLGİ            | ADRES | BİLGİ            | ADRES | BİLGİ            |
|-------|------------------|-------|------------------|-------|------------------|
|       | <b>R</b>         |       | <b>S</b>         |       | <b>Ş</b>         |
| 1033  | A 0001 4001 008D | 10FE  | A 0001 4FFA 008D | 11C9  | A 0001 5FF3 008D |
|       | B 0001 408E 008D |       | B 0001 5087 008D |       | B 0001 6080 008D |
|       | C 0001 411B 008D |       | C 0001 5114 008D |       | C 0001 610D 008D |
|       | Ç 0001 41A8 008D |       | Ç 0001 51A1 008D |       | Ç 0001 619A 008D |
|       | D 0001 4235 008D |       | D 0001 522E 008D |       | D 0001 6227 008D |
|       | E 0001 42C2 008D |       | E 0001 52BB 008D |       | E 0001 62B4 008D |
|       | F 0001 434F 008D |       | F 0001 5348 008D |       | F 0001 6341 008D |
|       | G 0001 43DC 008D |       | G 0001 53D5 008D |       | G 0001 63CE 008D |
|       | Ğ 0001 4469 008D |       | Ğ 0001 5462 008D |       | Ğ 0001 645B 008D |
|       | H 0001 44F6 008D |       | H 0001 54EF 008D |       | H 0001 64E8 008D |
|       | I 0001 4583 008D |       | I 0001 557C 008D |       | I 0001 6575 008D |
|       | İ 0001 4610 008D |       | İ 0001 5609 008D |       | İ 0001 6602 008D |
|       | J 0001 469D 008D |       | J 0001 5696 008D |       | J 0001 668F 008D |
|       | K 0001 472A 008D |       | K 0001 5723 008D |       | K 0001 671C 008D |
|       | L 0001 47B7 008D |       | L 0001 57B0 008D |       | L 0001 67A9 008D |
|       | M 0001 4844 008D |       | M 0001 583D 008D |       | M 0001 6836 008D |
|       | N 0001 48D1 008D |       | N 0001 58CA 008D |       | N 0001 68C3 008D |
|       | O 0001 495E 008D |       | O 0001 5957 008D |       | O 0001 6950 008D |
|       | Ö 0001 49EB 008D |       | Ö 0001 59E4 008D |       | Ö 0001 69DD 008D |
|       | P 0001 4A78 008D |       | P 0001 5A71 008D |       | P 0001 6A6A 008D |
|       | R 0001 4B05 008D |       | R 0001 5AFE 008D |       | R 0001 6AF7 008D |
|       | S 0001 4B92 008D |       | S 0001 5B8B 008D |       | S 0001 6B84 008D |
|       | Ş 0001 4C1F 008D |       | Ş 0001 5C18 008D |       | Ş 0001 6C11 008D |
|       | T 0001 4CAC 008D |       | T 0001 5CA5 008D |       | T 0001 6C9E 008D |
|       | U 0001 4D39 008D |       | U 0001 5D32 008D |       | U 0001 6D2B 008D |
|       | Ü 0001 4DC6 008D |       | Ü 0001 5DBF 008D |       | Ü 0001 6DB8 008D |
|       | V 0001 4E53 008D |       | V 0001 5E4C 008D |       | V 0001 6E45 008D |
|       | Y 0001 4EE0 008D |       | Y 0001 5ED9 008D |       | Y 0001 6ED2 008D |
|       | Z 0001 4F6D 008D |       | Z 0001 5F66 008D |       | Z 0001 6F5F 008D |

Tablo 8

| ADRES | BİLGİ            | ADRES | BİLGİ            | ADRES | BİLGİ            |
|-------|------------------|-------|------------------|-------|------------------|
|       | <b>T</b>         |       | <b>U</b>         |       | <b>Ü</b>         |
| 1294  | A 0001 6FEC 008D | 135F  | A 0001 7FE5 008D | 142A  | A 0001 8FDE 008D |
|       | B 0001 7079 008D |       | B 0001 8072 008D |       | B 0001 906B 008D |
|       | C 0001 7106 008D |       | C 0001 80FF 008D |       | C 0001 90F8 008D |
|       | Ç 0001 7193 008D |       | Ç 0001 818C 008D |       | Ç 0001 9185 008D |
|       | D 0001 7220 008D |       | D 0001 8219 008D |       | D 0001 9212 008D |
|       | E 0001 72AD 008D |       | E 0001 82A6 008D |       | E 0001 929F 008D |
|       | F 0001 733A 008D |       | F 0001 8333 008D |       | F 0001 932C 008D |
|       | G 0001 73C7 008D |       | G 0001 83C0 008D |       | G 0001 93B9 008D |
|       | Ğ 0001 7454 008D |       | Ğ 0001 844D 008D |       | Ğ 0001 9446 008D |
|       | H 0001 74E1 008D |       | H 0001 84DA 008D |       | H 0001 94D3 008D |
|       | I 0001 756E 008D |       | I 0001 8567 008D |       | I 0001 9560 008D |
|       | İ 0001 75FB 008D |       | İ 0001 85F4 008D |       | İ 0001 95ED 008D |
|       | J 0001 7688 008D |       | J 0001 8681 008D |       | J 0001 967A 008D |
|       | K 0001 7715 008D |       | K 0001 870E 008D |       | K 0001 9707 008D |
|       | L 0001 77A2 008D |       | L 0001 879B 008D |       | L 0001 9794 008D |
|       | M 0001 782F 008D |       | M 0001 8828 008D |       | M 0001 9821 008D |
|       | N 0001 78BC 008D |       | N 0001 88B5 008D |       | N 0001 98AE 008D |
|       | O 0001 7949 008D |       | O 0001 8942 008D |       | O 0001 993B 008D |
|       | Ö 0001 79D6 008D |       | Ö 0001 89CF 008D |       | Ö 0001 99C8 008D |
|       | P 0001 7A63 008D |       | P 0001 8A5C 008D |       | P 0001 9A55 008D |
|       | R 0001 7AF0 008D |       | R 0001 8AE9 008D |       | R 0001 9AE2 008D |
|       | S 0001 7B7D 008D |       | S 0001 8B76 008D |       | S 0001 9B6F 008D |
|       | Ş 0001 7C0A 008D |       | Ş 0001 8C03 008D |       | Ş 0001 9BFC 008D |
|       | T 0001 7C97 008D |       | T 0001 8C90 008D |       | T 0001 9C89 008D |
|       | U 0001 7D24 008D |       | U 0001 8D1D 008D |       | U 0001 9D16 008D |
|       | Ü 0001 7DB1 008D |       | Ü 0001 8DAA 008D |       | Ü 0001 9DA3 008D |
|       | V 0001 7E3E 008D |       | V 0001 8E37 008D |       | V 0001 9E30 008D |
|       | Y 0001 7ECB 008D |       | Y 0001 8EC4 008D |       | Y 0001 9EBD 008D |
|       | Z 0001 7F58 008D |       | Z 0001 8F51 008D |       | Z 0001 9F4A 008D |

Tablo 9

| ADRES | BİLGİ            | ADRES | BİLGİ            | ADRES | BİLGİ            |
|-------|------------------|-------|------------------|-------|------------------|
|       | <b>V</b>         |       | <b>Y</b>         |       | <b>Z</b>         |
| 14F5  | A 0001 9FD7 008D | 15C0  | A 0001 AFD0 008D | 168B  | A 0001 BFC9 008D |
|       | B 0001 A064 008D |       | B 0001 B05D 008D |       | B 0001 C056 008D |
|       | C 0001 A0F1 008D |       | C 0001 B0EA 008D |       | C 0001 C0E3 008D |
|       | Ç 0001 A17E 008D |       | Ç 0001 B177 008D |       | Ç 0001 C170 008D |
|       | D 0001 A20B 008D |       | D 0001 B204 008D |       | D 0001 C1FD 008D |
|       | E 0001 A298 008D |       | E 0001 B291 008D |       | E 0001 C28A 008D |
|       | F 0001 A325 008D |       | F 0001 B31E 008D |       | F 0001 C317 008D |
|       | G 0001 A3B2 008D |       | G 0001 B3AB 008D |       | G 0001 C3A4 008D |
|       | Ğ 0001 A43F 008D |       | Ğ 0001 B438 008D |       | Ğ 0001 C431 008D |
|       | H 0001 A4CC 008D |       | H 0001 B4C5 008D |       | H 0001 C4BE 008D |
|       | I 0001 A559 008D |       | I 0001 B552 008D |       | I 0001 C54B 008D |
|       | İ 0001 A5E6 008D |       | İ 0001 B5DF 008D |       | İ 0001 C5D8 008D |
|       | J 0001 A673 008D |       | J 0001 B66C 008D |       | J 0001 C665 008D |
|       | K 0001 A700 008D |       | K 0001 B6F9 008D |       | K 0001 C6F2 008D |
|       | L 0001 A78D 008D |       | L 0001 B786 008D |       | L 0001 C77F 008D |
|       | M 0001 A81A 008D |       | M 0001 B813 008D |       | M 0001 C80C 008D |
|       | N 0001 A8A7 008D |       | N 0001 B8A0 008D |       | N 0001 C899 008D |
|       | O 0001 A934 008D |       | O 0001 B92D 008D |       | O 0001 C926 008D |
|       | Ö 0001 A9C1 008D |       | Ö 0001 B9B 008D  |       | Ö 0001 C9B3 008D |
|       | P 0001 AA4E 008D |       | P 0001 BA47 008D |       | P 0001 CA40 008D |
|       | R 0001 AADB 008D |       | R 0001 BAD4 008D |       | R 0001 CACD 008D |
|       | S 0001 AB68 008D |       | S 0001 BB61 008D |       | S 0001 CB5A 008D |
|       | Ş 0001 ABF5 008D |       | Ş 0001 BBEE 008D |       | Ş 0001 CBE7 008D |
|       | T 0001 AC82 008D |       | T 0001 BC7B 008D |       | T 0001 CC74 008D |
|       | U 0001 AD0F 008D |       | U 0001 BD08 008D |       | U 0001 CD01 008D |
|       | Ü 0001 AD9C 008D |       | Ü 0001 BD95 008D |       | Ü 0001 CD8E 008D |
|       | V 0001 AE29 008D |       | V 0001 BE22 008D |       | V 0001 CE1B 008D |
|       | Y 0001 AEB6 008D |       | Y 0001 BEAF 008D |       | Y 0001 CEA8 008D |
|       | Z 0001 AF43 008D |       | Z 0001 BF3C 008D |       | Z 0001 CF35 008D |

Tablo 10

## 5.2. Ön Bellek Kartına Bilgilerin Yüklenmesi

D000:0000 adresinden itibaren, harfler al register'ına yüklenir ve daha sonra stosb komutuyla ön bellek kartına yazılır. Adresler ise ax register'ına yüklenerek, stosw komutuyla ön bellek kartına yazılır.

Bu yükleme işleminin nasıl yapıldığını bir program parçası ile görelim.

```
push ds      ; programın başlangıç
push es      ; Adreslerinin saklanması
```

;.....Ön Bellek Kartına Bilgilerin Yüklenmesi.....

```
mov ax, 0D000h ; Kartın segment adresi
mov es, ax
mov di, 00000h ; Kartın offset adresi
```

;.....1.Harflerin yanına ikinci harflerin başlangıç adreslerinin yüklenmesi

```
mov al, 'A'    ; 1.harf ve
stosb          ; ikinci harflerinin
mov ax, 00057h ; başlangıç adresi
stosw          ; ön bellek kartına yüklenir.
```

```
mov al, 'B'    ; 2.harf ve ikinci
stosb          ; harflerinin başlangıç
mov ax, 00122h ; adresi
stosw          ; ön bellek kartına yüklenir.
```

```
.
.
.
.
.
```

```
mov al, 'Z'    ; 29. harf ve ikinci
stosb          ; harflerinin başlangıç
mov ax, 0168Bh ; adresi
stosw          ; ön bellek kartına yüklenir.
```

;.....2.Harflerin yanına anahtar dosyasındaki adreslerin yüklenmesi

```

mov al, 'A'      ; AA ile başlayan bölgenin başlangıç
stosb           ; adresinin
mov ax, 00000h   ; Yüksek baytının
stosw           ; ve
mov ax, 0008Dh   ; Düşük baytının
stosw           ; Ön bellek kartına yüklenmesi
mov ax, 0008Dh   ; AA ile başlayan bölgenin uzunluğunun
stosw           ; ön bellek kartına yüklenmesi
.
.
.
.
.
.
.
mov al, 'Z'      ; ZZ ile başlayan bölgenin başlangıç
stosb           ; adresinin
mov ax, 00001h   ; Yüksek baytının
stosw           ;
mov ax, 0CF35h   ; düşük baytının
stosw           ; ve
mov ax, 0008Dh   ; ZZ ile başlayan bölgenin uzunluğunun
stosw           ; ön bellek kartına yüklenmesi

```

Tablolarda belirtilen harf ve adres bilgileri tablonun başından itibaren, program yardımıyla ön bellek kartına yüklenir. Program çalışmaya başlar başlamaz ilk önce bu bilgiler ön bellek kartına yüklenir, sonra diğer işlemler yapılır.

## BÖLÜM VI

### ANAHTAR VE ANA DOSYANIN OLUŞTURULMASI

#### 6.1. Handle Dosya ve Kayıt Organizasyonu

Unix/Xenix sisteminde kullanılan bu teknik , MS-DOS altında program yazarlar tarafından daha fazla tercih edilmektedir. Uygulamaların derli toplu olması, yazıcı kafanın pozisyonu için getirilen kolaylık, hiyerarşik dosya yapısını tanıması gibi özellikleri bu tercihin belli başlı sebeplerindendir.

Örneğin,

C:\DOS\COMMAND.COM gibi.

Bir dosya açılacak yada ilk kez yaratılacaksa, bahis konusu dosya adının segment adresi DS'ye, offset adresi ise DX registerine gönderilir ve INT 21H servisi çağrılır. Bu teknik kullanılırken aynı anda açık tutulabilecek dosya adedi, CONFIG.SYS içindeki FILES tanımlaması ile yapılabilir. Kullandığımız fonksiyonlar topluca aşağıda gösterilmiştir

| FONKSİYON | İŞLEVİ                       |
|-----------|------------------------------|
| 3Ch       | Dosya yaratma                |
| 3Dh       | Dosya açma                   |
| 3Eh       | Dosya kapama                 |
| 42h       | Dosya göstergesini konumlama |
| 3Fh       | Dosyadan kayıt oku           |
| 40h       | Dosyaya kayıt yaz            |

Bu fonksiyonların kullanımını kısaca inceleyelim.

#### 3Ch Fonksiyonu

İşlevi: Yeni bir dosya yaratmak

Kullanımı: ah=3Ch

ds:dx= Dosya adı segment ve offset

cx= Dosya özelliği

00= normal

01=sadece okunabilir

02=Görünmez

04=sistem

int 21h

Dönüşte: Carry flag =Clear ise başarılı ve ax'de handle.

Carry flag= set ise başarısız ax'de error kodları

ax=3 ise path bulunamadı

ax=4 ise boş handle yok

ax=5 ise giriş hatalı

Dosya, disk üzerinde başarıyla düzenlenmişse fonksiyon dönüşünde dosyaya ait handle ax'de okunacaktır. Daha sonra bu handle değeri, okuma, yazma, dosya göstergesinin konumlanması ve dosya kapama gibi işlemler için kullanılacaktır.

### 3Dh Fonksiyonu

İşlevi: Mevcut bir dosyayı açmak

Kullanımı: ah=3Dh

ds:dx= Dosya adı segment ve offseti

al = Ulaşım modu

00=okuma

01=yazma

02=okuma/yazma

int 21h

Dönüşte: Carry flag =Clear ise işlem başarılı ve ax= file handle.

Carry flag= set ise işlem başarısız ve ax= hata kodu

ax=1 ise geçersiz fonksiyon numarası

ax=2 ise dosya bulunamadı

ax=3 ise path bulunamadı

ax=4 ise boş handle yok

ax=5 ise erişim hatalı

### 3Eh Fonsiyonu

İşlevi: Dosya kapamak

Kullanımı: ah=3Eh

bx= file handle

int 21h

Dönüşte: Carry flag=Clear ise işlem başarılı ve ax'de file handle.

Carry flag= set ise işlem başarısız ve ax'de hata kodu

ax=6 ise dosya açık değil yada yanlış handle kullanımı

### 42h Fonksiyonu

İşlevi: Okuyucu/yazıcı kafanın pozisyonlandırılması

Kullanımı: ah=42h

al= metod

00=Dosyanın başından

01=mevcut yerden

02=Dosyanın sonundan

itibaren kafayı pozisyonlandırır.

bx= handle

cx= pozisyonun yüksek sözcüğü (yeni pozisyonun yüksek adresi)

dx=pozisyonun alçak sözcüğü (yeni pozisyonun alçak adresi)

int 21h

Dönüşte: Carry flag =Clear işlem başarılı ve ax'de yeni pozisyonun düşük sözcüğü

Carry flag= set işlem başarısız ve ax'de hata kodu

ax=1 ise geçersiz fonksiyon numarası

ax=6 ise geçersiz handle yada dosya açık değil

Bu fonksiyon ile okuyucu/yazıcı kafa, dosya içinde istenen yere konumlandırılıyor. Her başarılı okuma/yazma sonunda, kafa okunan/yazılan bayt kadar ilerliyor.

### 3Fh Fonksiyonu

İşlev : Bir kaydın okunması

Kullanım : ah=3Fh

bx= handle

cx=okunacak bayt adedi

ds:dx= kaydın okuyacağı buffer

int 21h

Dönüşte : Carry flag=Clear ise işlem başarılı ve ax'de okunan bayt adedi

Carry flag=Set ise işlem başarısız ve ax'de hata kodu

ax=5 ise ulaşılamadı

ax=6 ise Dosya kapalı yada geçersiz handle

### 40h Fonksiyonu

İşlev : Bir kaydın yazılması  
 Kullanım : ah=40h  
           bx= handle  
           cx=yazılacak bayt adedi  
           ds:dx= yazılacak buffer'ın segment ve offset adresi  
           int 21h  
 Dönüşte : Carry flag=Clear ise işlem başarılı ve ax'de yazılan bayt  
           Carry flag=Set ise işlem başarısız ve ax'de hata kodu  
           ax=5 ise erişim hatası  
           ax=6 ise dosya kapalı yada geçersiz handle [2]

## 6.2. Anahtar ve Ana Dosyanın Yaratılması

Dosya yaratma işlemi için klavyeden FileName bufferına maksimum 12 karakterlik bir dosya ismi alınır. Daha sonra DOS'un 3Ch fonksiyonunu kullanarak istediğimiz isimde bir dosya yaratabiliriz

Dosya yaratan bir program örneği aşağıda gösterilmektedir.

Örnek,

```
IDEAL
DOSSEG
MODEL    small
STACK    256
```

```
filemax    EQU 0Dh    ; Dosya ismi 12 karakter
```

```
DATASEG
```

```
exitCode    db    0
m1          db    0Dh, 09h, 09h, 09h, 'Dosya ismi= ', 0
m2          db    0Dh, 09h, 09h, 09h, '1 adet Dosya yaratıldı..... ', 0
m3          db    0Dh, 09h, 09h, 09h, ' Dosya yaratılmadı.....! ', 0
FileName    db    filemax DUP (0), 0 ; Dosya ismi + null
handle      dw    ?
```

## CODESEG

;..... From STRIO.OBJ

EXTRN StrRead:proc, Strwrite:proc, Newline:proc

```

Start :      mov  ax, @data
             mov  ds, ax
             mov  es, ax
             call NewLine           ;
             mov  di, OFFSET m1     ;Ekрана 'Dosya ismi=' yaz
             call StrWrite          ;
             mov  di, OFFSET FileName
             mov  cx, filemax       ;Dosya ismini 12 karakter olarak belirle
             call StrRead           ;Dosya ismini klavyeden al
             mov  ah, 3Ch           ;DOS dosya yaratma fonksiyonu
             mov  dx, OFFSET FileName
             mov  cx, 00h           ;Normal dosya
             int  21h               ;yaratma
             jc   error
             mov  [handle], ax
             call NewLine
             mov  di, OFFSET m2 ;'1 adet dosya yaratıldı.....' mesajını
             call Strwrite          ; Ekрана yaz
             jmp  Exit

```

```

Error:      call NewLine
             mov  di, OFFSET m3 ; 'Dosya yaratılmadı .....! 'mesajını
             call Strwrite          ; Ekрана yaz

```

```

Exit:      mov  ah, 04Ch           ; programı bitir ve
             mov  al, [exitCode]   ; DOS ortamına
             int  21h               ; dön
             END  Start

```

### 6.3. Anahtar ve Ana Dosyaya Kayıt Yapılması

3Dh fonksiyonu ile KEY.DAT (Anahtar Dosya) ve MASTER.DAT (Ana Dosya) yazma işlemi için açılır. Bu dosyalara ait handle değerleri daha sonra dosyaların yazma işleminde kullanılmak üzere saklanır.

Kayıt yapılacak esas bilgi 100 karakter olarak klavyeden testString buffer'ına alınır. Kayıt için 10 harfli anahtar kelime, klavyeden key bufferına alınır. Anahtar kelimenin ilk üç harfi karşılaştırma yapılmak üzere aşağıda belirtildiği şekilde alınır.

```

mov  si, OFFSET key      ; Anahtarın 1.harfi
lods [key]                ; al'ye yükle
mov  di, OFFSET harf1    ; al'nin içeriğini
mov  [harf1], al          ; harf1'e yükle
cld                       ; si=si+1, di=di+1
lods [key]                ; Anahtarın 2.harfini al'ye yükle
mov  di, OFFSET harf2    ; al'nin içeriğinin
mov  [harf2], al          ; harf 2'ye yüklenmesi
cld                       ; si=si+1, di=di+1
lods [key]                ; Anahtarın 3.harfini al'ye yükle
mov  di, OFFSET harf3    ; al'nin içeriğinin
mov  [harf3], al          ; harf 3 'ye yüklenmesi

```

Daha sonra anahtarın1.harfinin karşılaştırma yöntemi ile sayı karşılığının tesbiti aşağıda belirtildiği gibi yapılır.

```

mov  si, OFFSET harf1    ; Anahtarın 1.harfi
lods [harf1]              ; bl'ye
mov  bl, al               ; yüklenir.

```

```

A22: cmp  bl, 'A'         ; bl'nin içeriği alfabenin 1.harfi ile karşılaştırılır.
jne  B22                  ; eşit değilse alfabenin 2.harfi ile karşılaştırılır.
mov  ax, 00000h           ; eşitse 1.harfin sayısal karşılığı "0" dir.
mov  [G], ax              ; G =1, G= G- 1, G= 0
jmp  k3                   ; Anahtarın 2.harfinin sayısal tesbitine git

```

```

B22: cmp bl, 'B' ; bl'nin içeriği alfabenin 2.harfi ile karşılaştırılır.
jne C22 ; eşit değilse alfabenin 3. harfi ile karşılaştırılır.
mov ax, 00001h ; eşitse 2. harfin sayısal karşılığı "2" dir
mov [G], ax ; G = 2, G=G-1, G=1
jmp k3 ; Anahtarın 3.harfinin sayısal tesbitine git
.
.

```

```

Z22: cmp bl, 'Z' ;bl'nin içeriğini alfabenin 29.harfi ile karşılaştırılır
mov ax, 0001Ch
mov [G], ax ; G=1Dh, G=G-1, G=1Ch

```

Aynı şekilde ikinci harf için (H-1) ve üçüncü harf için (I-1)'in sayısal değerleri saptanır.

MASTER.DAT dosyası için kayıt numarası;

$$\text{RecNo} = (G-1) * 349h + (H-1) * 1Dh + (I-1) + 1$$

$$\text{NecNo} = (G-1) * 841 + (H-1) * 29 + (I-1) + 1$$

formülü ile hesaplanır. Burada;

G=Anahtarın 1.harfinin sayısal değeri  
H=Anahtarın 2.harfinin sayısal değeri  
I =Anahtarın 3.harfinin sayısal değeri

RecNo= Kayıt numarasıdır.

MASTER.DAT dosyası için adres;

Address =RecNo \* RecSize

formülü ile bulunur.

Burada;

RecNo = Kayıt numarası  
RecSize= Kayıt uzunluğu  
Address= Anahtara karşılık gelen adres'dir.

Key alanına MASTER.DAT dosyasının adreslerinin yerleştirilmesi;

Klavyeden alınan anahtarın uzunluğu bulunur ve anahtarın bitişinden itibaren, hesaplanan MASTER.DAT adresinin yüksek ve alçak adres baytları düzeltildikten sonra key bufferindeki anahtarın hemen arkasına yüklenir. Daha sonra MASTER.DAT dosyasının dosya göstergesi hesaplanan adres değerine ayarlanır, bu adresten itibaren MASTER.DAT dosyasına 100 bayt'lik esas bilgi kaydedilir. İlk iki harfi aynı olan başka kayıt olup olmadığı sorulur varsa yine kaydedilecek bilgi ve anahtar kelime alınır. Anahtar kelimenin ilk üç harfine göre anahtara karşılık gelen adres hesaplanır.

Anahtar ve hesaplanan adres 8Dh uzunluğundaki BigBuffer'a yüklenir. Daha sonra kaydedilecek bilgi MASTER.DAT dosyasının dosya göstergesinin değeri ayarlandıktan sonra MASTER.DAT dosyasına kaydedilir. Tekrar iki harfi aynı olan anahtarla kayıt yapılıp yapılmayacağı sorulur, yapılacakca aynı işlem tekrar edilir. Yapılmayacaksa BigBuffer içinde bulunan bilgilerin anahtar dosyaya kayıt edilmesi gerekir. Bunun için kaydın yapılacağı adres ön bellek kartından sağlanır. Önce anahtarın 1.harfi yanında 2.harfin başlangıç adresi, 2.harfin yanında da ilk iki harfi anahtarla aynı olan bölgenin başlangıç adresi ön bellek kartından alınır. KEY.DAT dosyasının dosya göstergesi ayarlanarak 141 bayt (8Dh) uzunluğundaki BigBuffer'deki Anahtar+Adres bilgileri KEY.DAT dosyasına yazılır.

KEY.DAT ve MASTER.DAT dosyalarına kayıtlar, klavyeden esas bilginin girilmesi ve kayıt için anahtar kelimenin alınması şeklinde devam eder.

## BÖLÜM VII

### ANAHTAR DOSYASININ ALTERNATİF VE BINARY SEARCH METODU İLE OKUNMASI

#### 7.1. Anahtar Dosyadaki İlk iki Harfi Aynı Olan Kayıtların Bulunduğu Bölgenin Alternatif Algoritma ile Bulunması

Arama işleminde, aranılacak anahtar kelime büyük harf olarak klavyeden buffera alınır. Anahtar kelimenin 1. harfi bl registerı'na, 2. harf cl registerına yüklenir. Ön bellek kartının segment ve offset adresleri ilgili register'lere (di:si) yüklenir. Lodsb komutu ile birinci harf ön bellek kartından al registerı'na alınır. Bu harf bl registerındaki harfle karşılaştırılır. Birbirine eşit değilse si'nin değeri (offset adresi) iki bayt kaydırılarak

```
(mov ax, si
add ax, 02h ; si=si+2
mov si, ax ),
```

bir sonraki harf ön bellek kartından al registerına yüklenir ve anahtar kelimenin birinci harfinin bulunduğu bl registerı ile karşılaştırılır. Register'ların içindeki bilgi birbirine eşit değilse, offset adresi yine iki bayt kaydırılarak, bir sonraki harf yine ön bellek kartından al registerına yüklenir ve bl registerı ile karşılaştırılır. Bu işlem anahtarın 1.harfi bulununcaya kadar devam eder. Birinci harf bulunduğu, lodsw komutu ile ön bellek kartından ax registerına ikinci harflerin başlangıç adresi yüklenir.

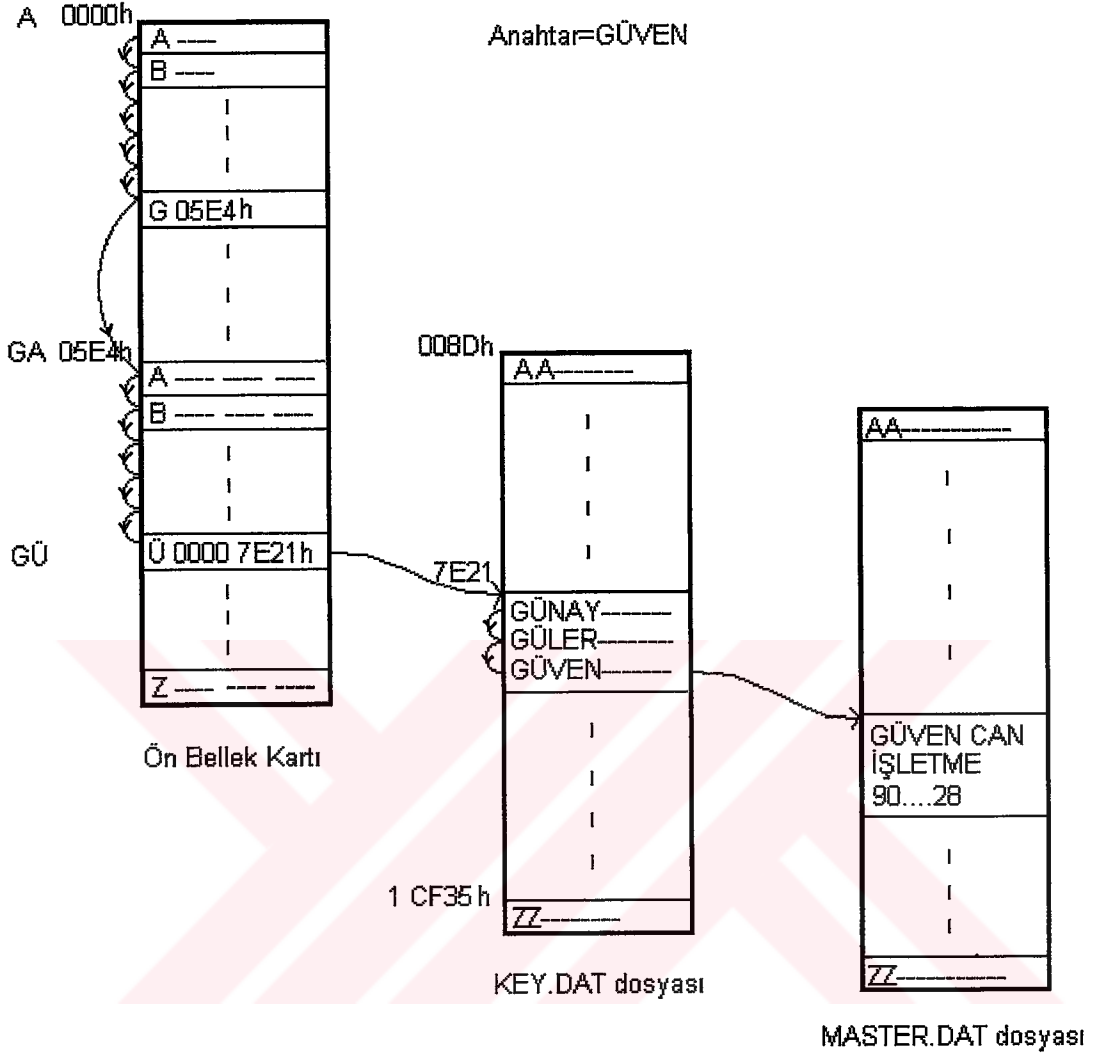
Bu adres bilgisi si registerına konularak bu adresten itibaren anahtar kelimenin ikinci harfi aranmaya başlanır. (al ve cl register'ları karşılaştırılır, birbirine eşit değilse offset adresi 6 bayt kaydırılarak, bir sonraki harf yine ön bellek kartından al registerına yüklenir ve cl registerı ile karşılaştırılır. Bu işlem anahtarın 2. harfi bulununcaya kadar devam eder). İkinci harf bulunduğu, lodsw komutu ile ikinci harfin yanındaki adresin yüksek 2 baytı'na ulaşılmış olunur ve bu adres bilgisi cx registerına yüklenir. Yine lodsw komutu ile bulunan adresin yanındaki adresin alçak 2 baytına ulaşılır. Bu adres bilgiside dx registerına yüklenir.

Yüksek adres baytları cx'de ve alçak adres baytları dx'te bulunduğundan, 42h fonksiyonunu kullanarak anahtar dosyanın file pointerı bu adrese ayarlanır. Daha sonra 3Fh fonksiyonu kullanarak 141 bayt'lık (8Dh) anahtar ve adres bilgileri bu yeni pozisyondan itibaren okunur.

Okunan blok içinde anahtarın tamamı StrPos modülü ile bulunur. Modülün çalışması için si'ye anahtarın adresi, di'ye taramanın yapılacağı hedef dizisinin (BigBuffer'in) adresi yüklenir. BigBuffer içinde karşılaştırma yöntemi ile anahtarın tamamı aranır. Anahtar bulunduğu zaman anahtar kelimenin indeksi dx registerındadır. dx'ın içeriği ax'e yüklenerek push komutu ile saklanır. Anahtarın yanındaki adres bilgilerine ulaşmak için anahtarın uzunluğu StrLength modülü (dizi içinde sıfır olmayan karakterlerin sayılması yöntemi) ile bulunur. Anahtarın uzunluğu cx registerındadır. Pop ax komutu ile anahtar kelimenin indeksi ax registerına yüklenmiş olur. ax ile cx toplanarak adreslerin indeksi tesbit edilir (anahtarın arkasındaki adres bilgisinin BigBuffer lokasyonunun başlangıcından itibaren kaç bayt ötede olduğu belirlenir).

Bu bilgi si register'ına yüklendikten sonra lodsw komutu ile MASTER.DAT dosyasının düşük adres baytları (2 bayt) ax register'ına yüklenir ve ax registeri içindeki bilgide data segmentte bulunan MADRESL lokasyonuna yüklenir. Sonra lodsw komutu ile MASTER.DAT dosyasının yüksek adres baytları (2 bayt) ax registerına alınır. Adresin düzeltilmesi için ax registerının içindeki değerin 30h, 31h ve 32h olup olmadığına bakılır. Eğer bu adreslerden birine eşitse ax registerından 30h çıkarılarak esas adres bilgisi elde edilir. Bu bilgide MADRESH lokasyonuna yüklenir. Adres düzeltme işlemine gerek duyulmasının nedeni 1h ve 2h değerlerinin aynı ASCII kodu ile gösterilmesinden kaynaklanmaktadır (aynı düzeltme kayıt sırasında da yapılıyor, yüksek adres değeri 0h, 1h ve 2h ise 30h ile toplanıp öyle kaydedilmektedir).

MADRESH lokasyonundaki adres bilgisi cx ve MADRESL lokasyonundaki adres bilgiside dx'e yüklendikten sonra, 42h fonksiyonu ile MASTER.DAT dosyasının file pointerı bu adrese ayarlanır. Daha sonra 3Fh fonksiyonu kullanılarak ayarlanan bu yeni pozisyondan itibaren 100 bayt'lık bir bilgi okunarak ekrana yazılır. Böylece okunmak istenen esas bilgiye erişmiş oluruz.



Şekil 7.1. Alternatif Algoritma ile İçinde Anahtarın Tamamının Aranılacağı Bölgenin Bulunması

KEY.DAT dosyasının okuması işleminde Alternatif Algoritmayı kullanan bir program örneği aşağıda gösterilmiştir.

%TITLE " ALTERNATİF ARAMA ALGORİTMASI ve ARAMA SÜRESİNİN SAPTANMASI 25.7.1997 Tekirdağ "

IDEAL  
DOSSEG  
MODEL small  
STACK 256

|         |          |                                                  |
|---------|----------|--------------------------------------------------|
| MaxLen  | EQU 65h  | ; 100-character string                           |
| cr      | EQU 13   | ; ASCII carriage return                          |
| lf      | EQU 10   | ; ASCII line feed                                |
| filemax | EQU 0Dh  | ; Filename 12 character                          |
| keymax  | EQU 0Bh  | ; key 10 character                               |
| keybuf  | EQU 0Fh  | ; kaydedilecek bilgi 14 byte                     |
| keybuf1 | EQU 0Bh  |                                                  |
| big     | EQU 08Dh | ; 141 Byte KEY.DAT'tan bir defada okunacak bilgi |
| T       | EQU 0Fh  | ; KEY.DAT'tan oku.kayıt uzunluğu                 |
| can0    | EQU 0Ch  |                                                  |

### DATASEG

|            |    |                                                             |  |
|------------|----|-------------------------------------------------------------|--|
| exitCode   | db | 0                                                           |  |
| dgoster1   | db | 0Dh,09h,09h,09h,' KEY.DATın file pointerı ayarlanmadı..!',0 |  |
| dgoster2   | db | 0Dh,09h,' MASTER.DAT ın file pointerı ayarlanmadı..!',0     |  |
| acilmadi   | db | 0Dh,09h,' MASTER.DAT dosyası açılmadı.....!',0              |  |
| mkayityok  | db | 0Dh,09h,09h,' Bu isimde kayıt yok.....!',0                  |  |
| testString | db | MaxLen DUP (?) ; MaxLen chars + null                        |  |
| KeyFileN   | db | 'KEY.DAT',0 ; Anahtar Dosya ismi                            |  |
| FileName   | db | 'MASTER.DAT',0                                              |  |
| handle     | dw | ?                                                           |  |
| handle1    | dw | ?                                                           |  |
| say        | dw | ?                                                           |  |
| key        | db | keybuf DUP (?) ; Aranılacak alt dizi                        |  |
| key1       | db | keybuf1 DUP (?)                                             |  |
| BigBuffer  | db | big DUP (?) ; Key'in aranılacağı buffer                     |  |
| target     | db | T DUP (0), 0 ; KEY.DAT'tan okunacak kayıt alanı             |  |
| harf1      | db | 1 DUP (0), 0 ; 1.karakter alanı                             |  |
| harf2      | db | 1 DUP (0), 0 ; 2.karakter alanı                             |  |
| harf3      | db | 1 DUP (0), 0 ; 3.karakter alanı                             |  |
| mkey       | db | 0Dh,09h,' Kayıt için anahtar kelimeyi yazınız: ',0          |  |
| mkey1      | db | 0Dh,09h,' Okuma için anahtar kelimeyi yazınız: ',0          |  |
| RecSize    | dw | 0065h ; Ana dosya için kayıt uzn.100 byte (64+Null=65h)     |  |

|        |    |   |                                                |
|--------|----|---|------------------------------------------------|
| ADRESL | dw | ? | ; Anahtara karş. gelen düşük adresin sak. alan |
| ADRESH | dw | ? | ; Anahtara karş. gelen yüksek adresin sak alan |

|         |    |   |                                     |
|---------|----|---|-------------------------------------|
| MADRESL | dw | ? | ; MASTER.DAT'in düşük adr.sak.alan  |
| MADRESH | dw | ? | ; MASTER.DAT'in yüksek adr.sak.alan |

|         |    |   |                                                     |
|---------|----|---|-----------------------------------------------------|
| MAXAREA | dw | ? | ; Anahtar kelimenin aranılacağı alanın max.uzunluğu |
|---------|----|---|-----------------------------------------------------|

|         |    |                 |
|---------|----|-----------------|
| saatbuf | db | can0 DUP (0), 0 |
| bolen   | db | 0Ah             |
| saat    | dw | ?               |

```

dakika    dw    ?
saniye    dw    ?
salise    dw    ?

```

## CODESEG

```
;----- From STRIO.OBJ:
```

```
    EXTRN    StrRead:proc, StrWrite:proc, NewLine:proc
```

```
;----- From STRING.OBJ:
```

```
    EXTRN StrInsert:proc, StrPos:proc, StrNull:proc, StrLength:proc
```

```
Start:    mov    ax, @data
           mov    ds, ax
           mov    es, ax
```

```
           push   ds ; Programın başlangıç
           push   es ; adresini sakla
```

```
;===== Ön belleğe bilginin yüklenmesi -----
```

```
           mov    ax, 0D000h ; Bellek kartının yer. adresin
           mov    es, ax      ; segment adresini
           mov    di, 00000h ; offset adresini yükle
```

```
;----- 1.karakterin yanına ikinci karakterin baş.adresinin yüklenmesi
```

```
           mov    al, 'A'
           stosb
           mov    ax, 00057h
           stosw
```

```
           mov    al, 'B'
           stosb
           mov    ax, 00122h
           stosw
```

```
           mov    al, 'C'
           stosb
           mov    ax, 001EDh
           stosw
```

```

.
.

```

.  
.  
.  
.

```
mov  al, 'Z'
stosb
mov  ax, 0168Bh
stosw
```

;===== A\* ikinci harfin yanındaki bilginin yüklenmesi

```
mov  al, 'A'
stosb
mov  ax, 00000h
stosw
mov  ax, 0008Dh
stosw
mov  ax, 0008Dh
stosw
```

.  
.  
.  
.

```
mov  al, 'Z'
stosb
mov  ax, 00000h
stosw
mov  ax, 00FF9h
stosw
mov  ax, 0008Dh
stosw
```

;===== B\* ikinci harfin yanındaki bilginin yüklenmesi

```
mov  al, 'A'
stosb
mov  ax, 00000h
stosw
mov  ax, 01086h
stosw
mov  ax, 0008Dh
stosw
```

.  
.  
.  
.

```
mov  al, 'Z'
stosb
```

```

mov ax, 00000h
stosw
mov ax, 01FF2h
stosw
mov ax, 0008Dh
stosw
.
.
.
.

```

;===== Z\* ikinci harfin yanındaki bilginin yüklenmesi

```

mov al, 'A'
stosb
mov ax, 00001h
stosw
mov ax, 0BFC9h
stosw
mov ax, 0008Dh
stosw
.
.
.
.
mov al, 'Z'
stosb
mov ax, 00001h
stosw
mov ax, 0CF35h
stosw
mov ax, 0008Dh
stosw

pop es
pop ds

```

;===== Dosyadan Bilgi Okuma işleminin Başlangıcı =====

;----- Okuma için anahtar kelimenin alınması -----

```

mov di, OFFSET key1
call StrNull

call NewLine
mov di, OFFSET mkey1 ; 'Okuma için anahtar kelimeyi
call StrWrite         ; yazın' mesajı

```

```

mov  di, OFFSET key1    ; Anahtar kelimenin
mov  cx, keymax          ; 10 bayt
call StrRead            ; klavyeden alınması

```

\*\*\*\*\*

----- Arama'nın Başlangıç Zamanı'nın Alınması -----

\*\*\*\*\*

```

mov  ah, 2Ch            ;ch=saat, cl=dakika, dh=saniye, dl=salise
int  21h                ;hex olarak GET CURRENT TIME

```

----- Saat ve Dakika Değerlerinin ASCII'ye Çevrilmesi

```

mov  bx, cx             ;bh=saat, bl=dakika
mov  cl, 8
rol  bx, cl             ;bh=dakika, bl=saat

```

```

mov  al, bl
cmp  al, 00Ah
jl   Yerles0

```

```

mov  ah, 000h           ;ah=0
div  [bolen]            ;ah=kalan, al=bölüm
or   al, 030h           ;ASCII'ye
or   ah, 030h           ;çevir

```

```

mov  [saat], ax
jmp  Dakika0

```

```

Yerles0:  mov  ah, 000h           ;ah=0
          or   ax, 03030h        ;ASCII'ye çevir
          mov  cl, 8
          rol  ax, cl
          mov  [saat], ax

```

```

Dakika0:  mov  cl, 8
          rol  bx, cl           ;bh=saat, bl=dakika
          mov  al, bl
          cmp  al, 00Ah
          jl   Yerles01

```

```

          mov  ah, 000h         ;ah=0
          div  [bolen]          ;ah=kalan, al=bölüm
          or   al, 030h         ;ASCII'ye
          or   ah, 030h         ;çevir

```

```

          mov  [dakika], ax

```

```
jmp Saniye0
```

```
Yerles01:  mov  ah, 000h
           or   ax, 03030h ;ASCII'ye çevir
           mov  cl, 8
           rol  ax, cl
           mov  [dakika], ax
```

;----- Saniye ve Salise Değerlerinin ASCII'ye çevrilmesi

```
Saniye0:   mov  bx, dx      ;bh=saniye, bl=salise
           mov  cl, 8
           rol  bx, cl     ;bh=salise, bl=saniye
```

```
           mov  al, bl
           cmp  al, 00Ah
           jl   Yerles00
```

```
           mov  ah, 000h
           div  [bolen]    ;ah=kalan, al=bölüm
           or   al, 030h   ;ASCII'ye
           or   ah, 030h   ;çevir
```

```
           mov  [saniye], ax
           jmp  Salise0
```

```
Yerles00:  mov  ah, 000h   ;ah=0
           or   ax, 03030h
           mov  cl, 8
           rol  ax, cl
           mov  [saniye], ax
```

```
Salise0:   mov  cl, 8
           rol  bx, cl     ;bh=saniye, bl=salise
           mov  al, bl
           cmp  al, 00Ah
           jl   Yerles011
```

```
           mov  ah, 000h
           div  [bolen]    ;ah=kalan, bl=bölüm
           or   al, 030h   ;ASCII'ye
           or   ah, 030h   ;çevir
```

```
;
;           mov  cl, 8
;           rol  ax, cl    ;dec çevir
           mov  [salise], ax
           jmp  Ekran
```

```
Yerles011: mov  ah, 000h   ;ah=0
           or   ax, 03030h
```

```

mov  cl, 8
rol  ax, cl
mov  [salise], ax

```

;----- ASCII'ye Çevrilen Zaman Değerlerinin Buffer'a Yüklenmesi

```

Ekran:  mov  di, OFFSET saatbuf
        mov  ax, [saat]
        stosw                      ;saat
        cld

        mov  al, 03Ah              ; (:)
        stosb
        cld

        mov  ax, [dakika]
        stosw                      ;dakika
        cld

        mov  al, 03Ah              ; (:)
        stosb
        cld

        mov  ax, [saniye]
        stosw                      ;saniye
        cld

        mov  al, 02Ch              ; (,)
        stosb
        cld

        mov  ax, [salise]
        stosw                      ;salise

```

;----- Düzenlenen Buffer'ın Ekran Yazdırılması

```

call  NewLine

mov  di, OFFSET saatbuf
call  StrWrite

```

```

.*****
;
;  ----- Aramanın Başlangıç Zamanı Ekran Yazıldı -----

```

;----- Anahtarın 1. ve 2. karakterlerinin bl ve cl'ye yüklenmesi

```

mov  si, OFFSET key1      ; Anahtarın 1.karakterini
lods [key1]               ; BL'ye yükle
mov  bl, al

```

```

cld                      ; si<--si+1, di<--di+1
lods [key1]              ; Anahtarın 2.karakterini
mov  cl, al              ; CL'ye yükle

```

#### ;----- 1.Karakterin bulunması -----

```

push  ds                ; Programın başlangıç
push  es                ; adresinin saklanması

mov  ax, 0D000h         ; Ön bellek kartının segment adresi
mov  ds, ax
mov  si, 00000h         ; Ön bellek kartının offset adresi

aq:   lodsb              ; Ön bellekten 1 baytlık karakter bilgi. okunması
      cmp  al, bl        ; Anah.1.karak. ile okunan karakteri karşılaştır
      je   found1
      mov  ax, si        ; Ön bellekten okunacak bir sonraki karakter
      add  ax, 00002h    ; için adresin belirlenmesi
      mov  si, ax        ; Yeni offset adresinin yüklenmesi
      jmp  aq

found1: lodsw            ;2. karakterlerin başlangıç adresinin bulunması
        mov  si, ax      ;Offset adresinin yüklenmesi

```

#### ;----- 2.Karakterin bulunması -----

```

bq:   lodsb              ; Anahtarın 2.karakteri ile ön bellekten okunan
      cmp  al, cl        ; karakterlerin karşılaştırılması
      je   found2
      mov  ax, si        ; Ön bellekten okunacak bir sonraki karakter
      add  ax, 00006h    ; için, offset adresinin belirlenmesi
      mov  si, ax        ; Yeni offset adresinin yüklenmesi
      jmp  bq

found2: lodsw
        mov  cx, ax      ; KEY.DAT'ın yüksek adresinin cx'e yüklenmesi
        cld
        lodsw
        mov  dx, ax      ; KEY.DAT'ın alçak adresinin dx'e yüklenmesi
        cld
        lodsw
        mov  bx, ax      ; Aramanın yapılacağı bellek alanının uzunluğu

        pop  es
        pop  ds

mov  [ADRESH], cx        ; KEY.DAT'ın yüksek adresinin saklanması
mov  [ADRESL], dx        ; KEY.DAT'ın alçak adresinin saklanması
mov  [MAXAREA], bx       ; Aramanın yapılacağı bel.al.uz.saklanması

```

;----- KEY.DAT dosyasının Okunması -----

```

mov dx, OFFSET KeyFileN ; KEY.DAT dosyasının
mov ah, 03Dh ; okuma/yazma için
mov al, 00 ;2 ; açılması
int 21h
; jc out2error
mov [handle], ax

```

adres111:;----- File Pointer'in Ayarlanması -----

```

mov ax, [ADRESL] ; Düşük adresin
mov dx, ax ; dx'e yüklenmesi
mov ax, [ADRESH] ; Yüksek adresin
mov cx, ax ; cx'e yüklenmesi

mov ah, 042h ; DOS seek-file (dosya-araştırma fonk.)
mov al, 0 ; Dosyanın başlangıcından araştır
mov bx, [handle] ; Assign file handle to bx
int 21h
jmp tyoku
jc handerror1
jmp tyoku
handerror1: mov di, OFFSET dgoster1
call StrWrite

```

;----- Yeni Pozisyonda KEY.DAT Dosyasından 141 bayt'ın okunması-----

```

tyoku: mov di, OFFSET BigBuffer
call StrNull

mov ah, 3Fh ; DOS Dosya-okuma fonksiyonu
mov bx, [handle] ; BX'e dosya mandalının tanıtılması
mov cx, 0008Dh ; Okunacak bilgi 141 byte
mov dx, OFFSET BigBuffer ; Gidilecek adres
int 21h

call NewLine ; Okunan bloğu ekrana yaz
mov di, OFFSET BigBuffer
call StrWrite

```

;----- Okunan Blok İçinde Anahtarın Tamamının Bulunması -----

```

mov si, OFFSET key1 ; Aranılacak alt dizinin baş.adresi
mov di, OFFSET BigBuffer ; İçinde aramanın yap.dizinin baş ad.
call StrPos ; Aramanın yapılması

```

```

jz    foundkey
jmp   notfound

```

```
foundkey:  push  dx
```

```

;*****
;
;  ----- Anahtarın Bulunduğu zamanın Alınması -----
;

```

```

mov    ah, 2Ch      ;ch=saat, cl=dakika, dh=saniye, dl=salise
int     21h         ;hex olarak GET CURRENT TIME

```

```
;----- Saat ve Dakika Değerlerinin ASCII'ye Çevrilmesi
```

```

mov    bx, cx      ;bh=saat, bl=dakika
mov    cl, 8
rol    bx, cl      ;bh=dakika, bl=saat

```

```

mov    al, bl
cmp    al, 00Ah
jl     Yerles09

```

```

mov    ah, 000h    ;ah=0
div    [bolen]     ;ah=kalan, al=bölüm
or     al, 030h    ;ASCII'ye
or     ah, 030h    ;çevir

```

```

mov    [saat], ax
jmp    Dakika09

```

```

Yerles09:  mov    ah, 000h    ;ah=0
or     ax, 03030h    ;ASCII'ye çevir
mov     cl, 8
rol     ax, cl
mov     [saat], ax

```

```

Dakika09:  mov    cl, 8
rol     bx, cl      ;bh=saat, bl=dakika
mov     al, bl
cmp     al, 00Ah
jl     Yerles19

```

```

mov    ah, 000h    ;ah=0
div    [bolen]     ;ah=kalan, al=bölüm
or     al, 030h    ;ASCII'ye
or     ah, 030h    ;çevir

```

```

mov    [dakika], ax
jmp    Saniye09

```

```

Yerles19:  mov  ah, 000h
           or   ax, 03030h ;ASCII'ye çevir
           mov  cl, 8
           rol  ax, cl
           mov  [dakika], ax

```

;----- Saniye ve Salise Değerlerinin ASCII'ye çevrilmesi

```

Saniye09:  mov  bx, dx      ;bh=saniye, bl=salise
           mov  cl, 8
           rol  bx, cl      ;bh=salise, bl=saniye

```

```

           mov  al, bl
           cmp  al, 00Ah
           jl   Yerles009

```

```

           mov  ah, 000h
           div  [bolen]     ;ah=kalan, al=bölüm
           or   al, 030h    ;ASCII'ye
           or   ah, 030h    ;çevir

```

```

           mov  [saniye], ax
           jmp  Salise09

```

```

Yerles009:  mov  ah, 000h    ;ah=0
           or   ax, 03030h
           mov  cl, 8
           rol  ax, cl
           mov  [saniye], ax

```

```

Salise09:   mov  cl, 8
           rol  bx, cl      ;bh=saniye, bl=salise
           mov  al, bl
           cmp  al, 00Ah
           jl   Yerles0119

```

```

           mov  ah, 000h
           div  [bolen]     ;ah=kalan, bl=bölüm
           or   al, 030h    ;ASCII'ye
           or   ah, 030h    ;çevir

```

```

           mov  [salise], ax
           jmp  Ekran9

```

```

Yerles0119: mov  ah, 000h    ;ah=0
           or   ax, 03030h
           mov  cl, 8
           rol  ax, cl
           mov  [salise], ax

```

;----- ASCII'ye Çevrilen Zaman Değerlerinin Buffer'a Yüklenmesi

```
Ekran9:      mov     di, OFFSET saatbuf
              mov     ax, [saat]
              stosw
              cld
              ;saat

              mov     al, 03Ah
              stosb
              cld
              ; (:)

              mov     ax, [dakika]
              stosw
              cld
              ;dakika

              mov     al, 03Ah
              stosb
              cld
              ; (:)

              mov     ax, [saniye]
              stosw
              cld
              ;saniye

              mov     al, 02Ch
              stosb
              cld
              ; (.)

              mov     ax, [salise]
              stosw
              ;salise
```

;----- Düzenlenen Buffer'ın Ekranı Yazdırılması

```
call     NewLine

mov     di, OFFSET saatbuf
call     StrWrite
```

\*\*\*\*\*  
;  
;----- Anahtarın Bulunduğu Zamanı Ekranı Yazıldı -----

```
pop     dx

mov     di, OFFSET BigBuffer
mov     ax, di
add     ax, dx
push    ax
```

```

mov  di, OFFSET key1
call StrLength
pop  ax
add  ax, cx

```

```

mov  si, OFFSET Bigbuffer ; Anahtardan sonra gelen ADH'in
mov  si, ax                ; adresi
lodsw                ; MASTER.DAT için
mov  [MADRESL], ax       ; Düşük adres bilgisinin alınması

```

;----- Adresi düzelt

```

lodsw                ; MASTER.DAT için
cmp  ah, 030h        ; Yüksek adres bilgisinin
je   ci1              ; alınması
cmp  ah, 031h        ; ah=30h, 31h, 32h ise
je   ci1              ; ah=ah-30h yap
cmp  ah, 032h
je   ci1
jmp  ci              ; değilse aynı kalsın
ci1: sub  ah, 030h

```

```

ci:  cmp  al, 030h    ; al=30h, 31h, 32h ise
je   ci2              ; al=al-30h yap
cmp  al, 031h
je   ci2
cmp  al, 032h
je   ci2
jmp  ci0              ; değilse aynı kalsın
ci2: sub  al, 030h    ; al=al-30h

```

```

ci0: mov  [MADRESH], ax ; Alçak adres bilgisinin alınması
jmp  qqqq9

```

;----- KEY.DAT Dosyasının Kapatılması -----

```

notfound: call NewLine          ;Kayıt yok mesajını ekrana
mov  di, OFFSET mkayityok      ;yaz
call StrWrite                   ;

```

```

mov  ah, 3Eh              ;KEY.DAT dosyasını
mov  bx, [handle]         ;kapat
int  21h
jmp  Exit

```

qqqq9: ;----- MASTER.DAT Dosyasını Okuma İçin Açma -----

```

mov  dx, OFFSET FileName

```

```

mov  ah, 03Dh
mov  al, 00
int  21h
jc   out22error
mov  [handle1], ax
jmp  ayar99

```

```

out22error:  mov  di, OFFSET acilmadi
              call StrWrite
              jmp  Exit

```

;----- MASTER.DAT Dosyası için File Pointer,ın Ayarlanması --

```

ayar99:      mov  ax, [MADRESL]      ; Düşük adresin
              mov  dx, ax             ; dx'e yüklenmesi
              mov  ax, [MADRESH]    ; Yüksek adresin
              mov  cx, ax             ; cx'e yüklenmesi

              mov  ah, 042h          ; DOS seek-file (dosya-araştırma fonk.)
              mov  al, 0              ; Dosyanın başlangıcından araştır
              mov  bx, [handle1]      ; Assign file handle to bx
              int  21h
              jc   handerror2
              jmp  oku88

```

```

handerror2:  mov  di, OFFSET dgoster2
              call StrWrite
              jmp  Exit

```

```

oku88:      mov  ax, 00000h
              mov  [MADRESL], ax
              mov  [MADRESH], ax

```

;----- Dosyadan bilgi okuma -----

```

mov  di, OFFSET testString
call StrNull

mov  ah, 3Fh
mov  bx, [handle1]
mov  cx, 65h                ; Okunacak bilgi 100 byte
mov  dx, OFFSET testString
int  21h

call NewLine                ; Kayıtın ekrana
mov  di, OFFSET testString  ; yazdırılması
call StrWrite

mov  ah, 3Eh

```

```

mov  bx, [handle1]
int  21h

```

```

Exit:  mov  ah, 04Ch      ; DOS fonksiyonu: Programdan çıkış
      mov  al, [exitCode] ; Dönüş çıkış kodu değeri
      int  21h          ; Programdan DOS ortamına dönüş

      END  Start        ; Programın sonu

```

## 7.2. Anahtar Dosyasındaki İlk İki Harfi Aynı Olan Kayıtların Bulunduğu Bölgenin Binary Search Algoritması İle Bulunması

KEY.DAT dosyası 841 eşit kısma bölündüğü için her bir bölümü kaydetmek içinde bir kayıt numarası gerekmektedir. Bu nedenle 1'den 841'e kadar kayıt numaraları kullanılmıştır.

İlk kayıtın numarası 1, son kayıtın numarası 841'dir. İlk arama için, ilk kayıt numarası ile son kayıt numarası toplanıp ikiye bölündüğünde orta değeri bulunur. Bu değer anahtar dosyasının tam ortasındaki kayıt numarasına eşittir. Kayıt uzunluğu ile çarpıldığında ortadaki kayıtın adresi bulunur. Bu adresten itibaren bölgenin tamamı okunur. Okunan bölgenin ilk iki bayt'ı ile anahtar kelimenin ilk iki harfi ile karşılaştırılır. Birbirine eşitse arama sona erer. Bulunan orta değeri anahtardan büyükse, üst=orta-1 yapılır, alt değer sabit kalır. Eğer bulunan orta değeri anahtardan küçükse üst değeri sabit kalır, alt=orta+1 yapılır. Sonra bu yeni alt ve üst değerleri toplanıp yeniden ikiye bölünerek, yeni bir orta değeri bulunur. Yeni orta değerine göre hesaplanan adresten bölgenin tamamı okunur. Okunan bölgenin ilk iki bayt'ı ile anahtar kelimenin ilk iki harfi karşılaştırılır. Eşit değilse yeni orta değeri hesaplanır. Bu işlem dosyadan okunan bölgenin ilk iki bayt'ı ile anahtarın ilk iki harfi birbirine eşit olana kadar devam eder. Birbirine eşit olduğunda ise bölgenin içinde anahtarın tamamı aranarak bulunur. Anahtar bulunduğunda yanındaki adres bilgisinden ana dosyadaki istenilen kayıt'a ulaşılır.

Bu işlemi şu şekilde formülüze edebiliriz;

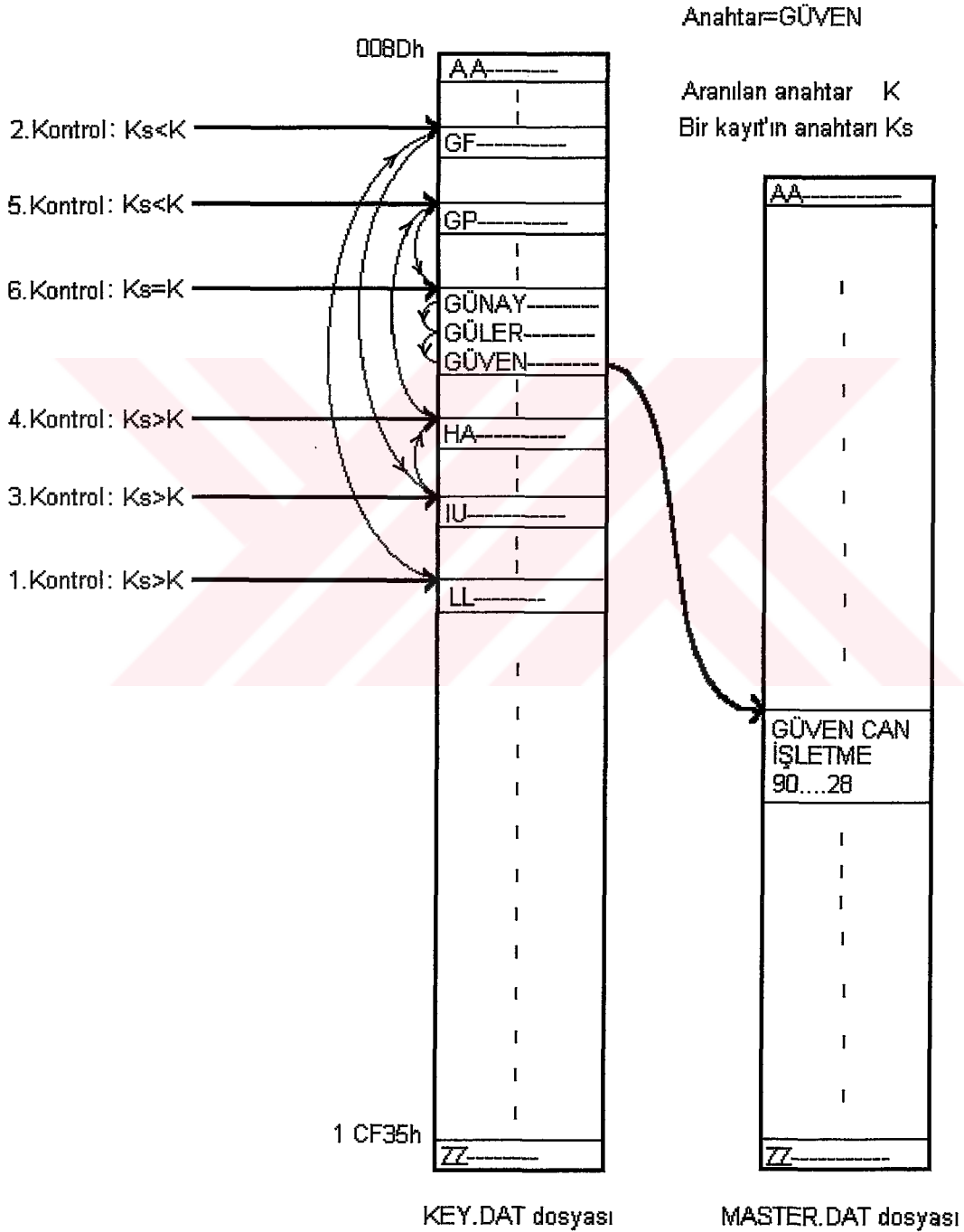
```

alt= 1      orta= (üst+alt) / 2      , orta=kayıt no
Üst=841     orta= anahtar ise arama sona erer

```

orta > anahtar ise üst=orta-1, alt=sabit  
 orta < anahtar ise üst=sabit, alt=orta+1 yapılır.

Sonra tekrar bölme ve karşılaştırma işlemi yapılır. Bu işlem orta=anahtar olana kadar devam eder.



Şekil 7.2. Binary Search Algoritması ile İçinde Anahtarın Tamamının Aranılacağı Bölgenin bulunması

KEY.DAT dosyasının okunması işleminde Binary Search Algoritmasını kullanan bir program örneği aşağıda gösterilmiştir.

%TITLE " BINARY SEARCH ARAMA ALGORİTMASI ve ARAMA SÜRESİNİN SAPTANMASI 15.8.1997 Tekirdağ "

```
IDEAL
DOSSEG
MODEL    small
STACK    256
```

```
keymax    EQU    0Bh
keybuf    EQU    0Fh
big       EQU    08Dh
buf       EQU    65h
can       EQU    0Ch
```

```
DATASEG
```

```
myok      db  0Dh,09h,09h,'Bu isimde bir kayıt yok.....!',0
okuma     db  0Dh,09h,09h,'Başka Okunacak Kayıt varmı E / H = ',0
mkey1     db  0Dh,09h,'B.SEARCH Okuma için anahtar kelimeyi yazınız = ',0
KeyFileN  db  'KEY.DAT',0
FileName  db  'MASTER.DAT',0
key       db  keybuf DUP (0), 0
key1      db  keybuf DUP (0), 0
```

```
exitCode  db  0
saatbuf   db  can DUP (0), 0
bolen     db  0Ah
saat      dw  ?
dakika    dw  ?
saniye    dw  ?
salise    dw  ?
```

```
handle    dw  ?
handle1   dw  ?
```

```
harf1     db  ?
harf2     db  ?
harf11    db  ?
harf22    db  ?
```

```
ust       dw  ?
```

```
alt       dw  ?
```

orta            dw    ?

top            dw    ?

keyh           dw    ?  
keyl           dw    ?

KADRESH      dw    ?  
KADRESL      dw    ?

MADRESH      dw    ?  
MADRESL      dw    ?

BigBuffer      db    big DUP (?)  
testString     db    buf DUP (?)  
buf1           dw    ?  
buf2           dw    ?

## CODESEG

;----- STRIO.OBJ'den

EXTRN   StrRead:proc, StrWrite:proc, NewLine:proc

;----- STRING.OBJ'den

EXTRN   StrLength:proc, StrRemove:proc, StrPos:proc, StrCompare:proc,  
         StrDelete:proc

EXTRN   StrNull:proc

Start:        mov    ax, @data  
              mov    ds, ax  
              mov    es, ax

;----- DOSYA'NIN ALT VE ÜST SINIRLARININ BELİRTİLMESİ -----

              mov    ax, 00349h  
              mov    [ust], ax

              mov    ax, 00001h  
              mov    [alt], ax

;----- Okuma için anahtar kelimenin alınması -----

qqqqqqq9:    mov    di, OFFSET key1  
              call   StrNull

```

call    NewLine
mov     di, OFFSET mkey1
call    StrWrite

```

```

mov     di, OFFSET key1
mov     cx, keymax
call    StrRead

```

\*\*\*\*\*

;----- Arama'nın Başlangıç Zamanı'nın Alınması

```

mov     ah, 2Ch    ;ch=saat, cl=dakika, dh=saniye, dl=salise
int     21h        ;hex olarak GET CURRENT TIME

```

;----- Saat ve Dakika Değerlerinin ASCII'ye Çevrilmesi

```

mov     bx, cx    ;bh=saat, bl=dakika
mov     cl, 8
rol     bx, cl    ;bh=dakika, bl=saat

```

```

mov     al, bl
cmp     al, 00Ah
jl      Yerles0

```

```

mov     ah, 000h    ;ah=0
div     [bolen]     ;ah=kalan, al=bölüm (decimale çevir)
or      al, 030h    ;ASCII'ye
or      ah, 030h    ;çevir
mov     [saat], ax
jmp     Dakika0

```

```

Yerles0: mov     ah, 000h    ;ah=0
or      ax, 03030h    ;ASCII'ye çevir
mov     cl, 8
rol     ax, cl
mov     [saat], ax

```

```

Dakika0: mov     cl, 8
rol     bx, cl        ;bh=saat, bl=dakika
mov     al, bl
cmp     al, 00Ah
jl      Yerles01

```

```

mov     ah, 000h    ;ah=0
div     [bolen]     ;ah=kalan, al=bölüm (decimale çevir)
or      al, 030h    ;ASCII'ye
or      ah, 030h    ;çevir

```

```

mov [dakika], ax
jmp Saniye0

```

```

Yerles01:  mov ah, 000h
           or  ax, 03030h ;ASCII'ye çevir
           mov cl, 8
           rol ax, cl
           mov [dakika], ax

```

;----- Saniye ve Salise Değerlerinin ASCII'ye çevrilmesi

```

Saniye0:   mov bx, dx      ;bh=saniye, bl=salise
           mov cl, 8
           rol bx, cl      ;bh=salise, bl=saniye

```

```

           mov al, bl
           cmp al, 00Ah
           jl  Yerles00

```

```

           mov ah, 000h
           div [bolen]      ;ah=kalan, al=bölüm (decimale çevir)
           or  al, 030h      ;ASCII'ye
           or  ah, 030h      ;çevir

```

```

           mov [saniye], ax
           jmp Salise0

```

```

Yerles00:  mov ah, 000h      ;ah=0
           or  ax, 03030h
           mov cl, 8
           rol ax, cl
           mov [saniye], ax

```

```

Salise0:   mov cl, 8
           rol bx, cl      ;bh=saniye, bl=salise
           mov al, bl
           cmp al, 00Ah
           jl  Yerles011

```

```

           mov ah, 000h
           div [bolen]      ;ah=kalan, bl=bölüm (decimale çevir)

```

```

           or  al, 030h      ;ASCII'ye
           or  ah, 030h      ;çevir

```

```

           mov [salise], ax
           jmp Ekran

```

```
Yerles011:  mov  ah, 000h  ;ah=0
            or   ax, 03030h
            mov  cl, 8
            rol  ax, cl
            mov  [salise], ax
```

;----- ASCII'ye Çevrilen Zaman Değerlerinin Buffer'a Yüklenmesi

```
Ekran:      mov  di, OFFSET saatbuf
            mov  ax, [saat]
            stosw                                ;saat
            cld

            mov  al, 03Ah                        ; (:)
            stosb
            cld

            mov  ax, [dakika]
            stosw                                ;dakika
            cld

            mov  al, 03Ah                        ; (:)
            stosb
            cld

            mov  ax, [saniye]
            stosw                                ;saniye
            cld

            mov  al, 02Ch                        ; (,)
            stosb
            cld

            mov  ax, [salise]
            stosw                                ;salise
```

;----- Düzenlenen Buffer'ın Ekrana Yazdırılması

```
            call NewLine
            mov  di, OFFSET saatbuf
            call StrWrite
```

\*\*\*\*\*

;----- Anahtarın ilk iki karakterinin karşılaştırma için alınması -----

```
            mov  si, OFFSET key1
            lods [key1]
```

```

mov  di, OFFSET harf1
mov  [harf1], al

cld
lods [key1]
mov  di, OFFSET harf2
mov  [harf2], al

bol:  mov  ah, 3Dh           ; KEY.DAT dosyasını
      mov  al, 02           ; okuma/yazma için
      mov  dx, OFFSET KeyFileN ; aç
      int  21h
      mov  [handle], ax

```

```

*****
;
;   BINARY SEARCH YÖNTEMİ İLE ARAMA YAPILACAK SAHA'NIN
;   ADRESİNİN BULUNMASI
;
*****
;
;   ORTA = (ÜST + ALT) / 2 , orta=RecNo
;
;   ORTA=KEY ise; aramaya son ver
;
;   ORTA > KEY ise; ÜST = ORTA - 1 , ALT = ALT
;
;   ORTA < KEY ise; ÜST = ÜST , ALT = ORTA + 1
;
*****
;

```

```

mov  ax, [ust]      ; bol:
add  ax, [alt]      ; (üst + alt)
sub  dx, dx
mov  bx, 00002h     ;
div  bx             ; (üst + alt) / 2 = ax

mov  [orta], ax     ;RecNo

mov  bx, 0008Dh     ;RecSize
mul  bx             ;Address=RecNo*RecSize

mov  [KADRESH], dx  ;Çarpımın yüksek seviyeli word'ünü sak.
mov  [KADRESL], ax  ;Çarpımın düşük seviyeli word'ünü sakla

```

;----- File Pointer'ın Ayarlanması -----

```

mov  cx, [KADRESH]  ;
mov  dx, [KADRESL]  ;

```

```

mov  ah, 042h
mov  al, 0
mov  bx, [handle]
int  21h

```

;----- Yeni Pozisyonda KEY.DAT dosyasından kayıt okunması -----

```

tyoku1:  mov  di, OFFSET BigBuffer      ; Anahtarın tamamının
        call StrNull                  ; içinde aranılacağı
        mov  ah, 3Fh                  ; bellek sahasının
        mov  bx, [handle]             ; KEY.DAT dosyasından
        mov  cx, 0008Dh              ; okunması
        mov  dx, OFFSET BigBuffer    ;
        int  21h

```

;----- Orta değerle Anahtar'ın değerinin karşılaştırılması -----

```

mov  si, OFFSET BigBuffer
lods  [BigBuffer]
mov  di, OFFSET harf11
mov  [harf11], al

```

```

cld
lods  [BigBuffer]
mov  di, OFFSET harf22
mov  [harf22], al

```

```

mov  ah, 3Eh      ;
mov  bx, [handle] ;KEY.DAT dosyasını kapat
int  21h      ;

```

```

mov  ah, [harf11]
mov  al, [harf22]
mov  bh, [harf1]
mov  bl, [harf2]

```

```

cmp  ax, bx
je   esit1
jmp  duzelt

```

```

esit1: jmp  esit11

```

;----- Karşılaştırma için yüksek baytları düzeltme

```

duzelt:  cmp  bh, 043h
        je   aa7
        cmp  bh, 047h
        je   aa7
        cmp  bh, 049h
        je   aa7

```

```

        cmp    bh, 04Fh
        je     aa7
        cmp    bh, 053h
        je     aa7
        cmp    bh, 055h
        je     aa7
        jmp    karx

aa7:    cmp    ah, 080h    ;Ç=D
        je     a1
        cmp    ah, 0A6h    ;Ĝ=H
        je     a2
        cmp    ah, 098h    ;İ=J
        je     a3
        cmp    ah, 099h    ;Ö=P
        je     a4
        cmp    ah, 09Eh    ;Ş=T
        je     a5
        cmp    ah, 09Ah    ;Ü=V
        je     a6
        jmp    kar

a1:     sub    ah, 03Ch
        jmp    kar
a2:     sub    ah, 05Eh
        jmp    kar
a3:     sub    ah, 04Eh
        jmp    kar
a4:     sub    ah, 049h
        jmp    kar
a5:     sub    ah, 04Ah
        jmp    kar
a6:     sub    ah, 044h
        jmp    kar

karx:   cmp    ah, 080h    ;Ç=C
        je     c
        cmp    ah, 0A6h    ;Ĝ=G
        je     g
        cmp    ah, 098h    ;İ=I
        je     i
        cmp    ah, 099h    ;Ö=O
        je     o
        cmp    ah, 09Eh    ;Ş=S
        je     s
        cmp    ah, 09Ah    ;Ü=U
        je     u
        jmp    kar0

```

```

c:      sub    ah, 03Dh
        jmp    kar
g:      sub    ah, 05Fh
        jmp    kar
i:      sub    ah, 04Fh
        jmp    kar
o:      sub    ah, 04Ah
        jmp    kar
s:      sub    ah, 04Bh
        jmp    kar
u:      sub    ah, 045h

kar:    cmp    bh, 080h    ;Ç=C
        je     ccc
        cmp    bh, 0A6h    ;Ĝ=G
        je     ggg
        cmp    bh, 098h    ;İ=I
        je     iii
        cmp    bh, 099h    ;Ö=O
        je     ooo
        cmp    bh, 09Eh    ;Ş=A
        je     ss00
        cmp    bh, 09Ah    ;Ü=U
        je     uuu
        jmp    duzelt0

ccc:    sub    bh, 03Dh
        jmp    duzelt0
ggg:    sub    bh, 05Fh
        jmp    duzelt0
iii:    sub    bh, 04Fh
        jmp    duzelt0
ooo:    sub    bh, 04Ah
        jmp    duzelt0
ss00:   sub    bh, 04Bh
        jmp    duzelt0
uuu:    sub    bh, 045h
        jmp    duzelt0

kar0:   cmp    ah, 044h
        je     kar
        cmp    ah, 048h
        je     kar
        cmp    ah, 04Ah
        je     kar
        cmp    ah, 050h
        je     kar
        cmp    ah, 054h
        je     kar

```

```

        cmp    ah, 056h
        je     kar

        cmp    bh, 080h    ;Ç=D
        je     c9
        cmp    bh, 0A6h    ;Ğ=H
        je     g9
        cmp    bh, 098h    ;İ=J
        je     i9
        cmp    bh, 099h    ;Ö=P
        je     o9
        cmp    bh, 09Eh    ;Ş=T
        je     s9
        cmp    bh, 09Ah    ;Ü=V
        je     u9
        jmp    duzelt0

c9:      sub    bh, 03Ch
        jmp    duzelt0
g9:      sub    bh, 05Eh
        jmp    duzelt0
i9:      sub    bh, 04Eh
        jmp    duzelt0
o9:      sub    bh, 049h
        jmp    duzelt0
s9:      sub    bh, 04Ah
        jmp    duzelt0
u9:      sub    bh, 044h

```

;————— Düşük baytları karşılaştırma için düzeltme

duzelt0:

```

        cmp    bl, 043h
        je     aa77
        cmp    bl, 047h
        je     aa77
        cmp    bl, 049h
        je     aa77
        cmp    bl, 04Fh
        je     aa77
        cmp    bl, 053h
        je     aa77
        cmp    bl, 055h
        je     aa77
        jmp    karxx

aa77:    cmp    ah, 080h    ;Ç=D
        je     a11

```

```

        cmp    ah, 0A6h    ;Ğ=H
        je     a22
        cmp    ah, 098h    ;İ=J
        je     a33
        cmp    ah, 099h    ;Ö=P
        je     a44
        cmp    ah, 09Eh    ;Ş=T
        je     a55
        cmp    ah, 09Ah    ;Ü=V
        je     a66
        jmp    kar5

a11:    sub     ah, 03Ch
        jmp    kar5
a22:    sub     ah, 05Eh
        jmp    kar5
a33:    sub     ah, 04Eh
        jmp    kar5
a44:    sub     ah, 049h
        jmp    kar5
a55:    sub     ah, 04Ah
        jmp    kar5
a66:    sub     ah, 044h
        jmp    kar5

karxx:  cmp     al, 080h    ;Ç=C
        je     c5
        cmp     al, 0A6h    ;Ğ=G
        je     g5
        cmp     al, 098h    ;İ=I
        je     i5
        cmp     al, 099h    ;Ö=O
        je     o5
        cmp     al, 09Eh    ;Ş=S
        je     s5
        cmp     al, 09Ah    ;Ü=U
        je     u5
        jmp    kar01

c5:     sub     al, 03Dh
        jmp    kar5
g5:     sub     al, 05Fh
        jmp    kar5
i5:     sub     al, 04Fh
        jmp    kar5
o5:     sub     al, 04Ah
        jmp    kar5
s5:     sub     al, 04Bh
        jmp    kar5

```

```

u5:      sub    al, 045h

kar5:    cmp    bl, 080h      ;Ç=C
         je     ccc5
         cmp    bl, 0A6h      ;Ĝ=G
         je     ggg5
         cmp    bl, 098h      ;I=I
         je     iii5
         cmp    bl, 099h      ;Ö=O
         je     ooo5
         cmp    bl, 09Eh      ;Ş=A
         je     sss5
         cmp    bl, 09Ah      ;Ü=U
         je     uuu5
         jmp    kar11

ccc5:    sub    bl, 03Dh
         jmp    kar11
ggg5:    sub    bl, 05Fh
         jmp    kar11
iii5:    sub    bl, 04Fh
         jmp    kar11
ooo5:    sub    bl, 04Ah
         jmp    kar11
sss5:    sub    bl, 04Bh
         jmp    kar11
uuu5:    sub    bl, 045h
         jmp    kar11

kar01:   cmp    al, 044h      ;al=D
         je     kar5
         cmp    al, 048h      ;al=H
         je     kar5
         cmp    al, 04Ah      ;al=J
         je     kar5
         cmp    al, 050h      ;al=P
         je     kar5
         cmp    al, 54h       ;al=T
         je     kar5
         cmp    al, 56h       ;al=V
         je     kar5

         cmp    bl, 080h      ;Ç=D
         je     c99
         cmp    bl, 0A6h      ;Ĝ=H
         je     g99
         cmp    bl, 098h      ;I=J
         je     i99
         cmp    bl, 099h      ;Ö=P

```

```

        je      o99
        cmp     bl, 09Eh      ;Ş=T
        je      s99
        cmp     bl, 09Ah      ;Ü=V
        je      u99
        jmp     kar11

c99:     sub     bl, 03Ch
        jmp     kar11
g99:     sub     bl, 05Eh
        jmp     kar11
i99:     sub     bl, 04Eh
        jmp     kar11
o99:     sub     bl, 049h
        jmp     kar11
s99:     sub     bl, 04Ah
        jmp     kar11
u99:     sub     bl, 044h

kar11:   cmp     ax, bx
        jg      compare1
        jmp     compare2

```

compare1: ;----- Orta>Key ise üst=orta-1, alt=alt=Sabit kalır

```

        mov     ax, [orta]      ;
        sub     ax, 01h         ; üst=orta-1
        mov     [ust], ax      ;
        jmp     bol

```

compare2: ;----- Orta<Key ise üst=üst=Sabit kalır, alt=orta+1

```

        mov     ax, [orta]
        add     ax, 01h
        mov     [alt], ax

        jmp     bol

```

esit11: ;===== Anahtarın tamamının bulunması =====

;----- KEY.DAT Dosyasını aç

```

        mov     ah, 3Dh
        mov     al, 02
        mov     dx, OFFSET KeyFileN
        int     21h
        mov     [handle], ax

```

;----- KEY.DAT'ın file pointerını ayarla

```
mov  cx, [KADRESH]
mov  dx, [KADRESL]
mov  ah, 042h
mov  al, 0
mov  bx, [handle]
int  21h
```

;----- KEY.DAT'tan buffer'a 008Dh bayt oku

```
mov  di, OFFSET BigBuffer
call StrNull

mov  ah, 3Fh
mov  bx, [handle]
mov  cx, 0008Dh
mov  dx, OFFSET BigBuffer
int  21h
```

;----- Buffer içinde anahtarın tamamını bulma

```
call NewLine
mov  di, OFFSET BigBuffer
call StrWrite

mov  si, OFFSET key1
mov  di, OFFSET BigBuffer
call StrPos
jz   foundkey
jmp  notfound
```

foundkey: push dx

\*\*\*\*\*

```
,
mov  ah, 2Ch      ;ch=saat, cl=dakika, dh=saniye, dl=salise
int  21h          ;hex olarak GET CURRENT TIME
```

;----- Saat ve Dakika Değerlerinin ASCII'ye Çevrilmesi

```
mov  bx, cx      ;bh=saat, bl=dakika
mov  cl, 8
rol  bx, cl      ;bh=dakika, bl=saat

mov  al, bl
cmp  al, 00Ah
jl   Yerles09
```

```

mov ah, 000h      ;ah=0
div [bolen]        ;ah=kalan, al=bölüm (decimale çevir)

```

```

or al, 030h        ;ASCII'ye
or ah, 030h         ;çevir

```

```

mov [saat], ax
jmp Dakika09

```

```

Yerles09: mov ah, 000h      ;ah=0
or ax, 03030h      ;ASCII'ye çevir
mov cl, 8
rol ax, cl
mov [saat], ax

```

```

Dakika09: mov cl, 8
rol bx, cl          ;bh=saat, bl=dakika
mov al, bl
cmp al, 00Ah
jl Yerles19

```

```

mov ah, 000h      ;ah=0
div [bolen]        ;ah=kalan, al=bölüm (decimale çevir)

```

```

or al, 030h        ;ASCII'ye
or ah, 030h         ;çevir

```

```

mov [dakika], ax
jmp Saniye09

```

```

Yerles19: mov ah, 000h      ;ah=0
or ax, 03030h      ;ASCII'ye çevir
mov cl, 8
rol ax, cl
mov [dakika], ax

```

;----- Saniye ve Salise Değerlerinin ASCII'ye çevrilmesi

```

Saniye09: mov bx, dx          ;bh=saniye, bl=salise
mov cl, 8
rol bx, cl          ;bh=salise, bl=saniye

```

```

mov al, bl
cmp al, 00Ah
jl Yerles009

```

```

mov ah, 000h
div [bolen]        ;ah=kalan, al=bölüm (decimale çevir)

```

```

or    al, 030h    ;ASCII'ye
or    ah, 030h    ;çevir

```

```

mov    [saniye], ax
jmp    Salise09

```

```

Yerles009: mov    ah, 000h    ;ah=0
or      ax, 03030h
mov     cl, 8
rol     ax, cl
mov     [saniye], ax

```

```

Salise09: mov     cl, 8
rol      bx, cl      ;bh=saniye, bl=salise
mov      al, bl
cmp      al, 00Ah
jl       Yerles0119

```

```

mov     ah, 000h
div     [bolen]      ;ah=kalan, bl=bölüm (decimale çevir)

```

```

or      al, 030h    ;ASCII'ye
or      ah, 030h    ;çevir

```

```

mov     [salise], ax
jmp     Ekran9

```

```

Yerles0119: mov    ah, 000h    ;ah=0
or      ax, 03030h
mov     cl, 8
rol     ax, cl
mov     [salise], ax

```

;————— ASCII'ye Çevrilen Zaman Değerlerinin Buffer'a Yüklenmesi

```

Ekran9:  mov     di, OFFSET saatbuf
mov      ax, [saat]
stosw                                ;saat
cld

```

```

mov     al, 03Ah    ; (:)
stosb
cld

```

```

mov     ax, [dakika]
stosw                                ;dakika
cld

```

```

mov     al, 03Ah    ; (:)

```

```

stosb
cld

mov ax, [saniye]
stosw          ;saniye
cld

mov al, 02Ch   ; (.)
stosb
cld

mov ax, [salise]
stosw          ;salise

```

;----- Düzenlenen Buffer'ın Ekrana Yazdırılması

```

call NewLine

mov di, OFFSET saatbuf
call StrWrite

```

\*\*\*\*\*

```

pop dx
mov di, OFFSET BigBuffer
mov ax, di
add ax, dx
push ax
mov di, OFFSET key1
call StrLength
pop ax
add ax, cx
mov si, OFFSET BigBuffer
mov si, ax
lodsw
mov [MADRESL], ax

```

;----- Adresi düzelt

```

lodsw
cmp ah, 030h
je ci1
cmp ah, 031h
je ci1
cmp ah, 032h
je ci1
jmp ci
ci1: sub ah, 030h

ci: cmp al, 030h
je ci2

```

```

        cmp     al, 031h
        je      ci2
        cmp     al, 032h
        je      ci2
        jmp     ci0
ci2:    sub     al, 030h
ci0:    mov     [MADRESH], ax

```

;=====MASTER.DAT DOSYASINI AÇ VE OKU=====

```

        mov     dx, OFFSET FileName
        mov     ah, 03Dh
        mov     al, 02
        int     21h
        mov     [handle1], ax

```

```

        mov     ax, [MADRESL]
        mov     dx, ax
        mov     ax, [MADRESH]
        mov     cx, ax
        mov     ah, 042h
        mov     al, 0
        mov     bx, [handle1]
        int     21h

```

```

        mov     ax, 00000h
        mov     [MADRESL], ax
        mov     [MADRESH], ax

```

;----- MASTER.DAT Dosyasından Bilgi Okuma

```

        mov     di, OFFSET testString
        call    StrNull

```

```

        mov     ah, 3Fh
        mov     bx, [handle1]
        mov     cx, 65h
        mov     dx, OFFSET testString
        int     21h

```

```

        call    NewLine
        mov     di, OFFSET testString
        call    Strwrite
        jmp     ex0

```

```

notfound: call    NewLine
          mov     di, OFFSET myok

          call    StrWrite

```

;----- Dosyayı kapama  
ex0:

```
    mov  ah, 3Eh  
    mov  bx, [handle]  
    int  21h
```

```
    mov  ah, 3Eh  
    mov  bx, [handle1]  
    int  21h
```

Exit: 

```
    mov  ah, 04Ch  
    mov  al, 000h  
    int  21h  
    END  Start
```



## BÖLÜM VIII

### BINARY SEARCH VE ALTERNATİF ALGORİTMANIN KARŞILAŞTIRILMASI

Binary Search metodu ile okuma yapabilmek için önce KEY.DAT dosyasındaki 841 bellek bölgesinin hepsine birer kayıt yapıldı (Bunu yapmamızın sebebi binary search metodu ile bulduğumuz adresten okuma yaptığımızda eğer kayıt yoksa, bulunan değer anahtarla karşılaştırma işlemi gerçekleşmediğinden program sonsuz döngüye giriyor).

Arama süresinin saptanmasında DOS kesmelerinden 2Ch fonksiyonundan yararlanılmıştır.

DOS - O anki sistem zamanını verir (GET Current Time )

AH = 2Ch

Dönen Değer:

CH = Saat hex olarak (0-17h =00-23d)

CL = Dakika hex olarak (0-3Bh=0-59d)

DH = Saniye hex olarak (0-3Bh=00-59d)

DL = Salise hex olarak (Saniyenin yüzdebiri, 0-63h=0-99d).

Anahtar kelime klavyeden girildikten sonra, o anki sistem zamanı ekrana yazılıyor ve okunan bellek bölgesi içinde anahtarın tamamı bulunduğunda ise o zamanki sistem zamanı ekrana yazılıyor. İki okuma arasındaki zaman farkı salise (saniyenin 100'de biri) olarak bize anahtara erişim süresini vermektedir. Binary Search ve Alternatif Algoritmanın karşılaştırılması için Bölüm VII de verilen örnek programlar kullanılmıştır.

Pentium-S 75MHz mikroişlemcili ve 256K ön bellekli bir bilgisayarla yapılan aramalar sonucu, anahtarlara erişim süreleri salise olarak tablo 11, tablo 12 ve tablo 13'te gösterilmiştir.

Tablolardaki değerler her bir anahtar için, bilgisayar resetlendikten sonra alınmıştır. Bunu yapmamızın nedeni birincil ve ikincil ön belleğin etkisini ortadan kaldırmaktır.

| Aranılan Anahtarlar | Alternatif | B. Search | Aranılan Anahtarlar | Alternatif | B.Search |
|---------------------|------------|-----------|---------------------|------------|----------|
| ABI                 | 0          | 5         | BALABAN             | 6          | 6        |
| ACAR                | 0          | 5         | BARIN               | 0          | 5        |
| ACI                 | 0          | 5         | BAŞ                 | 0          | 6        |
| AGIRBAŞ             | 0          | 6         | BAYRAKTAR           | 5          | 5        |
| AKALAN              | 0          | 5         | BEKTAŞ              | 5          | 5        |
| AKBABA              | 5          | 5         | BEŞİKTEPE           | 5          | 5        |
| AKCA                | 0          | 6         | BEYSÖZLÜ            | 0          | 6        |
| AKÇAKIL             | 5          | 11        | BİÇEN               | 0          | 5        |
| AKGÜN               | 5          | 4         | BİLGİ               | 0          | 5        |
| AKHAN               | 5          | 5         | BİRİNCİ             | 0          | 5        |
| ALHAN               | 0          | 5         | BÖLÜKBAŞI           | 6          | 6        |
| ALTINTAŞ            | 0          | 6         | BOYACIOĞLU          | 0          | 6        |
| ANAÇ                | 0          | 6         | BOZDAĞ              | 0          | 6        |
| ANIÇ                | 6          | 6         | CAN                 | 6          | 6        |
| APAYDIN             | 0          | 6         | CENGİZ              | 0          | 5        |
| ARACI               | 5          | 6         | CÖMERT              | 0          | 6        |
| ARIKAN              | 0          | 6         | ÇAĞDAŞ              | 0          | 6        |
| ARŞIK               | 5          | 6         | ÇAKIR               | 5          | 5        |
| ARYINDOĞAN          | 0          | 6         | ÇALIŞKAN            | 5          | 5        |
| AŞÇI                | 0          | 6         | ÇANKIR              | 5          | 5        |
| ATAMAN              | 6          | 6         | ÇELİK               | 5          | 5        |
| ATEŞ                | 5          | 5         | ÇETİN               | 0          | 5        |
| ATLI                | 5          | 5         | ÇİFTÇİ              | 0          | 6        |
| AVCI                | 5          | 5         | ÇORA                | 0          | 6        |
| AYAV                | 5          | 5         | DALGIN              | 5          | 5        |
| AYDIN               | 0          | 5         | DAYAN               | 0          | 6        |
| AYKUT               | 0          | 5         | DEMİR               | 0          | 5        |
| AYSAN               | 0          | 6         | DENİZ               | 0          | 6        |
| AYTAR               | 6          | 11        | DİKEL               | 6          | 5        |
| BABALIK             | 5          | 6         | DİNÇER              | 0          | 6        |
| BADEM               | 6          | 6         | DIYAROĞLU           | 0          | 5        |
| BAKAR               | 0          | 5         | DOĞAN               | 6          | 6        |

Tablo 11

| Aranılan Anahtarlar | Alternatif | B. Search | Aranılan Anahtarlar | Alternatif | B.Search |
|---------------------|------------|-----------|---------------------|------------|----------|
| DÖNMEZ              | 0          | 6         | HAYTAN              | 6          | 5        |
| DURAN               | 6          | 11        | IŞIK                | 5          | 11       |
| DÜZ                 | 0          | 5         | İMRAL               | 0          | 6        |
| ELBİR               | 11         | 16        | İNAN                | 6          | 6        |
| ELİUYGUN            | 11         | 16        | KAÇAN               | 5          | 6        |
| ELMAS               | 11         | 17        | KAHRAMAN            | 5          | 11       |
| EMRE                | 6          | 17        | KALKAN              | 0          | 6        |
| ER                  | 38         | 49        | KANDEMİR            | 6          | 6        |
| ERGÜN               | 44         | 50        | KARSLIOĞLU          | 5          | 11       |
| ERHAN               | 39         | 49        | KASIM               | 5          | 11       |
| ERKUŞ               | 38         | 44        | KAYA                | 5          | 5        |
| EROĞLU              | 39         | 49        | KAZAK               | 0          | 11       |
| ERSEN               | 38         | 44        | KELEBEK             | 0          | 6        |
| ERTEMİZ             | 44         | 49        | KESKİN              | 0          | 5        |
| EŞK                 | 6          | 16        | KOÇ                 | 6          | 6        |
| EYÜPOĞLU            | 5          | 11        | KOK                 | 5          | 6        |
| FERHATOĞLU          | 5          | 5         | KOL                 | 0          | 6        |
| FİDANBOY            | 5          | 5         | KONGÜR              | 5          | 6        |
| GARIP               | 5          | 5         | KOPARAN             | 0          | 6        |
| GAYRETLİ            | 0          | 11        | KÖPRÜLÜ             | 5          | 6        |
| GENÇ                | 0          | 11        | KÖRKUYU             | 0          | 6        |
| GİRGİN              | 0          | 5         | KÖSELER             | 0          | 5        |
| GÖKÇEK              | 5          | 5         | MARANGOZ            | 6          | 5        |
| GÖLEMEN             | 0          | 11        | MERAL               | 0          | 6        |
| GÖNENÇ              | 0          | 6         | NAYIR               | 0          | 6        |
| GÖRGÜLÜ             | 5          | 11        | OLCAY               | 5          | 6        |
| GÜLER               | 6          | 11        | ONAR                | 5          | 6        |
| GÜN                 | 6          | 6         | ÖNDER               | 0          | 6        |
| GÜVEN               | 5          | 11        | ÖRNEK               | 0          | 6        |
| HACET               | 5          | 6         | ÖZALP               | 0          | 11       |
| HANEY               | 5          | 11        | ÖZCAN               | 0          | 6        |
| HASDEMİR            | 5          | 11        | ÖZDEMİR             | 0          | 6        |

Tablo 12

| Aranılan Anahtarlar | Alternatif | B. Search | Aranılan Anahtarlar | Alternatif | B.Search |
|---------------------|------------|-----------|---------------------|------------|----------|
| ÖZEK                | 5          | 5         | TUNA                | 5          | 5        |
| ÖZGEN               | 5          | 5         | TÜZÜN               | 6          | 11       |
| PAŞA                | 0          | 11        | UÇAR                | 0          | 6        |
| PEKTAŞ              | 0          | 6         | ULAŞ                | 5          | 11       |
| SALDIR              | 6          | 6         | ULUSAN              | 6          | 11       |
| SARIOĞLU            | 5          | 6         | UYSAL               | 5          | 5        |
| SEÇER               | 0          | 6         | UZUN                | 5          | 6        |
| SELVİ               | 0          | 6         | ÜNAL                | 5          | 6        |
| SEMERÇİ             | 0          | 6         | ÜNER                | 6          | 11       |
| SERBEST             | 0          | 6         | ÜNGÖR               | 0          | 6        |
| SEVİNÇ              | 0          | 6         | ÜSTÜNER             | 0          | 11       |
| SEZER               | 0          | 6         | VAROL               | 0          | 5        |
| SÖĞÜT               | 0          | 6         | VATANSEVER          | 5          | 5        |
| SÖNMEZ              | 0          | 6         | VURAL               | 6          | 5        |
| SUCUOĞLU            | 0          | 6         | YAKAN               | 6          | 11       |
| SULUHAN             | 0          | 6         | YAMAN               | 6          | 11       |
| ŞAHİN               | 6          | 6         | YANAR               | 6          | 11       |
| ŞENER               | 6          | 6         | YARAR               | 6          | 11       |
| ŞİMŞEK              | 6          | 11        | YAVUZ               | 5          | 11       |
| ŞİŞMAN              | 5          | 11        | YAZAN               | 0          | 11       |
| TALI                | 5          | 11        | YELLİ               | 0          | 6        |
| TAMER               | 0          | 6         | YÜKSEL              | 6          | 11       |
| TAN                 | 5          | 11        | ZEYBEK              | 5          | 11       |
| TARÇIN              | 0          | 5         | Anahtar sayısı      | 183        | 183      |
| TAŞKIN              | 5          | 11        | Toplam süre         | 803        | 1582     |
| TEKİN               | 6          | 11        | Ortalama            | 4.38       | 8.64     |
| TEPEDÜZÜ            | 6          | 5         | Fark (8.64-4.38)    | 4.26       |          |
| TEZCAN              | 5          | 6         |                     |            |          |
| TOKER               | 0          | 5         |                     |            |          |
| TOPALOĞLU           | 0          | 5         |                     |            |          |
| TÖRE                | 0          | 11        |                     |            |          |
| TRAK                | 6          | 6         |                     |            |          |

Tablo 13

## BÖLÜM IX

### SONUÇ

Okuma sürelerinin tesbiti için yapılan kayıtlarda 183 anahtar kelime kullanılmıştır.

Anahtar kelimelerle yapılan okumalar esnasında makinanın kendi ön belleklerinin etkisini ortadan kaldırmak için, bütün okumalar makine reset edildikten sonra yapılmıştır.

Alternatif algoritma ile 183 anahtar toplam olarak 803 salise'de okunmuş ve anahtar dosya içindeki bir anahtara erişim süresi ortalama olarak 4.38 salise olarak bulunmuştur.

Binary Search algoritması kullanılarak aynı anahtarlar toplam olarak 1582 salise'de okunmuştur. Anahtar dosya içindeki bir anahtara erişim süresi ise ortalama olarak 8.64 salise olarak bulunmuştur.

Bu anahtarlarla yapılan tesbite göre, Alternatif algoritmamız bir anahtara ortalama olarak Binary Search algoritmasından 4.26 salise daha önce ulaşmaktadır.

**KAYNAKLAR**

- [1] INTEL, 1992, Microprocessors, Intel Corporation USA, Volum II, p.4.261
- [2] KILIÇLI, A.,D., 1995, Assembler Uygulamalarıyla PC'nin Sırları, Sistem Yayıncılık İstanbul, s.119
- [3] MARTIN, J., 1976 , Principles Of Data-Base-Management, Prentice-Hall Inc. Englewood Cliffs New Jersey, p.111
- [4] MESSMER, H., P., 1995, The Indispensable PC Hardware Book, Addison-Wesley Publishing Company Inc.,USA, p.37
- [5] OLCAY, N., 1993, A Cache-Independent Benchmark Suite, Boğaziçi Üniversitesi Yüksek Lisans Tezi, s.30
- [6] SWAN, T., 1991, Mastering Turbo Assembler, Hayden Books Indiana USA, pp.317-346
- [7] YILMAZ, M., 1995, PC Donanımı, Sistem Yayıncılık İstanbul, s.40

## ÖZGEÇMİŞ

- 26.02.1959 ..... İstanbul'da doğdum. İlkokul, Ortaokul ve Lise eğitimimi İstanbul'da tamamladım.
- 1982 ..... Tekirdağ Meslek Yüksekokulu'nun Elektronik Bölümünden bölüm birincisi olarak mezun oldum.
- 1985 ..... Dikey geçiş yaptığım Marmara Üniversitesi Teknik Eğitim Fakültesinin Elektronik Anabilim Dalını iyi derece ile bitirdim.
- 27.11.1986 ..... T.Ü.Tekirdağ Meslek Yüksekokulu'na Öğretim Görevlisi olarak atamam yapıldı.
- 14.12.1986-
- 30.07.1987 ..... Askerliğimi kısa dönem olarak tamamladım.
- 02.09.1987 ..... Tekrar eski görevime atandım. Halen Öğretim Görevlisi ve End.Elektronik Program Danışmanı olarak bu görevde bulunmaktayım.
- 19.11.1990-
- 19.01.1991 ..... YÖK/Dünya Bankası Endüstriyel Eğitimi projesi ile Amerika'da Ferris State Üniversitesinde Endüstriyel Elektronik Öğretmenliği Eğitimine katıldım.
- 09.08.1993 ..... Marmara Üniversitesi Fen Bilimleri Enstitüsü Elektronik-Bilgisayar Eğitimi Ana Bilim Dalında Yüksek Lisansımı yaptım ve iyi derece ile mezun oldum.

## EK-1

### Kullanılan Modül Programlar

#### 1. Dizi Yordamları

TITLE "String Procedures--Copyright 1989 by Tom Swan"

```
IDEAL
DOSSEG
MODEL    small
```

```
CODESEG
```

```
PUBLIC    MoveLeft, MoveRight, StrNull, StrLength
PUBLIC    StrUpper, StrCompare, StrDelete, StrInsert
PUBLIC    StrConcat, StrCopy, StrPos, StrRemove
```

---

```
; Assemble with the command TASM STRING to create STRING.OBJ. To use
; the procedures, Add EXTRN (procedure):PROC statements where
; (procedure) is one of the following identifiers:
```

```
;
; MoveLeft      --memory move with increasing indexes
; MoveRight     --memory move with decreasing indexes
; StrNull       --erase all chars in string
; StrLength     --return number of chars in string
; StrUpper      --convert chars in string to uppercase
; StrCompare    --alphabetically compare two strings
; StrDelete     --delete chars from string
; StrInsert     --insert chars into string
; StrConcat     --attach one string to another
; StrCopy       --copy one string to another
; StrPos        --find position of substring in a string
; StrRemove     --remove substring from a string
;
```

```
; After assembling your program, link with STRING.OBJ. For example,
; if your program is named MYPROG, first assemble MYPROG to
MYPROG.OBJ
; and link with the command TLINK MYPROG+STRING to create
MYPROG.EXE.
```

```
; STRING VARIABLES:
```

```
; A string is a simple array of characters with one character per
```

;8-bit byte. A null character (ASCII 0) must follow the last  
 ;character in the string. An empty string contains a single null.  
 ;Declare string variables this way:

```
;
;   STRING      DB   81 DUP (0) ; 80-character string+null
;
```

;STRING CONSTANTS:

;Always allow one extra byte for the null terminator. Character  
 ;constants (which may be used as variables) must be properly  
 ;terminated. For example:

```
;
;   C1      DB   'This is a test string.',0
;
```

;SEGMENT REGISTERS:

;Routines in this package assume that ES and DS address the  
 ;same segment. Set ES=DS before calling any of these routines.

```
ASCNull      EQU 0      ; ASCII null character
```

```
%NEWPAGE
```

```
;MoveLeft      Move byte-block left (down) in memory
```

;Input:

```
;   si=address of source string (s1)
;   di=address of destination string (s2)
;   bx=index s1 (i1)
;   dx=index s2 (i2)
;   cx=number of bytes to move (count)
```

;Output:

```
;   count bytes from s1 |i1| moved to the location
;   starting at s2|i2|
```

;Registers:

```
;   none
```

```
PROC      MoveLeft
jcxz     @@99      ;Exit if count=0
push     cx        ;Save modified registers
push     si
push     di

        add     si, bx      ;Index into source string
        add     di, dx      ;Index into destination string
        cld          ;Auto-increment si and di
        rep     movsb      ;Move while cx () 0

        pop     di        ;Restore registers
        pop     si
```

```

        pop    cx
@@99:
        ret                    ;Return to caller
ENDP    MoveLeft
%NEWPAGE

```

---

```

;MoveRight  Move byte-block right (up) in memory

```

---

```

;Input:
;    (same as MoveLeft)
;Output:
;    (same as MoveLeft)
;Registers:
;    none

```

---

```

PROC    MoveRight
        jcxz   @@99           ;Exit if count=0
        push  cx              ;Save modified registers
        push  di
        push  si

        add   si, bx          ;Index into source string
        add   di, dx          ;Index into destination string
        add   si, cx          ;Adjust to last source byte
        dec   si
        add   di, cx          ;Adjust to last destination byte
        dec   di
        std   si              ;Auto-decrement si and di
        rep   movsb           ;Move while cx () 0

        pop   si              ;Restore registers
        pop   di
        pop   cx

```

```

@@99:
        ret                    ;Return to caller
ENDP    MoveRight
%NEWPAGE

```

---

```

;StrNull      Erase all characters in string

```

---

```

;Input:
;    di=address of string (s)
;Output:
;    s[0] < - null character (ASCII 0)
;Registers:
;    none

```

---

```

PROC    StrNull
        mov    [byte ptr di], ASCNull    ;Insert null at s[0]

```

```

                ret                                ;Return to caller
ENDP           StrNull
%NEWPAGE

```

```

;StrLength      Count non-null characters in a string

```

```

;Input:

```

```

;      di=address of string (s)

```

```

;Output:

```

```

;      cx=number of non-null characters in s

```

```

;Registers:

```

```

;      cx

```

```

PROC           StrLength
                push ax                        ;Saved modified registers
                push di
                xor  al, al                    ;al < - search char (null)
                mov  cx, 0ffffh               ;cx - maximum search depth
                cld                             ;Auto-increment di
                repnz scasb                   ;Scan for al while [di] <> null & cx <>
                not  cx                       ;Ones complement of cx
                dec  cx                       ;minus 1 equals string length

                pop  di                       ;Restore registers
                pop  ax
                ret                             ;Return to caller
ENDP           StrLength
%NEWPAGE

```

```

;StrUpper          Convert chars in string to uppercase

```

```

;Input:

```

```

;      di=address of string to convert (s)

```

```

;Output:

```

```

;      lowercase chars in string converted to uppercase

```

```

;Registers:

```

```

;      none

```

```

PROC           StrUpper
                push ax                        ;Save modified registers
                push cx
                push di
                push si
                call StrLength                ;Set cx=length of string
                jcxz @@99                    ;Exit if length=0
                cld                             ;Auto-increment si, di
                mov  si, di                   ;Set si=di

```

```

@@10:

```

```

        lodsb                ;al < - s[si]; si < - si + 1
        cmp    al, 'a'       ;Is al >= 'a'?
        jnb    @@20          ;No, jump to continue scan
        cmp    al, 'z'       ;Is al <= 'z'?
        ja     @@20          ;No, jump to continue scan
        sub    al, 'a'-'A'    ;Convert lowercase to uppercase
@@20:
        stosb                ;s[di] < - al; di < - di + 1
        loop   @@10          ;cx < - cx -1; loop if cx <> 0
@@99:
        pop    si            ;Restore registers
        pop    di
        pop    cx
        pop    ax
        ret                 ;Return to caller
ENDP    StrUpper
%NEWPAGE

```

---

```

;StrCompare Compare two strings

```

---

```

;Input:

```

```

;    si=address of string 1 (s1)

```

```

;    di=address of string 2 (s2)

```

```

;Output:

```

```

;    flags set for conditional jump using jb, jbe,
;    je, ja, or jae.

```

```

;Registers:

```

```

;    none

```

---

```

PROC    StrCompare
        push    ax           ;Save modified registers
        push    di
        push    si
        cld                ;Auto-increment si
@@10:
        lodsb                ;al = |si|, si = si+1
        scasb                ;Compare al and |di|; di = di+1
        jne     @@20          ;Exit if nonequal chars found
        or      al, al        ;Is al=0? (i.e., at end of s1)
        jne     @@10          ;If no jump, else exit
@@20:
        pop     si           ;Restore registers
        pop     di
        pop     ax
        ret                 ;Return flags to caller
ENDP    StrCompare
%NEWPAGE

```

---

```
;StrDelete    Delete characters anywhere in a string
```

---

```
;Input:
```

```
;    di=address of string (s)
;    dx=index (i) of first char to delete
;    cx=number of chars to delete (n)
```

```
;Output:
```

```
;    n characters deleted from string at s[i]
;    Note: prevents deleting past end of string
```

```
;Registers:
```

```
;    none
```

---

```
PROC          StrDelete
```

```
    push  bx          ;Save modified registers
    push  cx
    push  di
    push  si
```

```
;bx=SourceIndex
```

```
;cx=Count/Len/CharsToMove
```

```
;dx=Index
```

```
    mov  bx, dx        ; Assign string index to bx
    add  bx, cx        ; Source index - index+count
    call StrLength     ; cx - length(s)
    cmp  cx, bx        ; Is length ) index?
    ja   @@10          ; If yes, jump to delete chars
    add  di, dx        ; else, calculate index to string end
    mov  [byte ptr di], ASCNull ; and insert null
    jmp  short @@99    ; Jump to exit
```

```
@@10:
```

```
    mov  si, di        ; Make source = destination
    sub  cx, bx        ; CharsToMove <- Len - SourceIndex
    inc  cx            ; Plus one for null at end of string
    call MoveLeft      ; Move chars over deleted portion
```

```
@@99:
```

```
    pop  si            ; Restore registers
    pop  di
    pop  cx
    pop  bx
    ret                ; Return to caller
```

```
ENDP          StrDelete
```

```
%NEWPAGE
```

---

```
;StrInsert    Insert a string into another string
```

---

```
;Input:
```

```
;    si = address of string 1 (s1)
```

```

; di = address of string 2 (s2)
; dx = insertion index for s2 (i)
; NOTE: s2 must be large enough to expand by length(s1)!
; Output:
;   chars from string s1 inserted at s2[i]
;   s1 not changed
; Registers:
;   none

```

---

```

PROC      StrInsert
    push  ax          ; Save modified registers
    push  bx
    push  cx

; ax = LenInsertion
; cx = CharsToMove

    xchg  si, di      ; Exchange si and di
    call  StrLength   ; and find length of s1
    xchg  si, di      ; Restore si and di
    mov   ax, cx      ; Save length(s1) in ax
    call  StrLength   ; Find length of s2
    sub   cx, dx      ; cx <- length(s2) - i + 1
    inc   cx          ; cx = (CharsToMove)

; bx = s1 index

    push  dx          ; Save index (dx) and si
    push  si
    mov   si, di      ; Make si and di address s2
    mov   bx, dx      ; Set s1 index to dx (i)
    add   dx, ax      ; Set s2 index to i + LenInsertion
    call  MoveRight   ; Open a hole for the insertion
    pop   si          ; Restore index (dx) and si
    pop   dx

    xor   bx, bx      ; Set s1 (source) index to 0
    mov   cx, ax      ; Set cx to LenInsertion
    call  MoveLeft    ; Insert s1 into hole in s2

    pop   cx          ; Restore registers
    pop   bx
    pop   ax
    ret              ; Return to caller
ENDP      StrInsert
%NEWPAGE

```

```

;-----
; StrConcat  Concatenate (join) two strings
;-----
; Input:
;   si = address of source string (s1)
;   di = address of destination string (s2)
;   Note: s2 must be large enough to expand by length (s1)!
; Output:
;   chars from s1 added to end of s2
; Registers:
;   none
;-----
PROC      StrConcat
    push  bx          ; Save modified registers
    push  cx
    push  dx

; dx = s2 destination

    call  StrLength    ; Find length of destination (s2)
    mov   dx, cx       ; Set dx to index end of string
    xchg  si, di       ; Exchange si and di
    call  StrLength    ; Find length of source (s1)
    inc   cx           ; Plus 1 includes null terminator
    xchg  si, di       ; Restore si and di
    xor   bx, bx       ; Source index = 0
    call  MoveLeft     ; Copy source string to destination

    pop   dx           ; Restore registers
    pop   cx
    pop   bx
    ret              ; Return to caller
ENDP      StrConcat
%NEWPAGE
;-----
; StrCopy    Copy one string to another
;-----
; Input:
;   si = address of source string (s1)
;   di = address of destination string (s2)
; Output:
;   Chars in s1 copied to s2
;   Note: s2 must be at least Length(s1) + 1 bytes long
; Registers:
;   none
;-----
PROC      StrCopy
    push  bx          ; Save modified registers
    push  cx

```

```

push dx

xchg si, di      ; Swap si and di
call StrLength   ; Find length of source string (s1)
inc cx           ; Plus 1 includes null terminator
xchg si, di      ; Restore si and di
xor bx, bx       ; Source-string index = 0
xor dx, dx       ; Destination-string index = 0
call MoveLeft    ; Copy source to destination

pop dx           ; Restore registers
pop cx
pop bx
ret              ; Return to caller
ENDP StrCopy
%NEWPAGE

```

---

```

; StrPos          Search for position of a substring in a string

```

---

```

; Input:
;   si = address of substring to find
;   di = address of target string to scan
; Output:
;   if zf = 1 then dx = index of substring
;   if zf = 0 then substring was not found
;   Note: dx is meaningless if zf = 0
; Registers:
;   dx

```

---

```

PROC StrPos
push ax          ; Save modified registers
push bx
push cx
push di

call StrLength   ; Find length of target string
mov ax, cx       ; Save length (s2) in ax
xchg si, di      ; Swap si and di
call StrLength   ; Find length of substring
mov bx, cx       ; Save length (s1) in bx
xchg si, di      ; Restore si and di
sub ax, bx       ; ax = last possible index
jb @@20          ; Exit if len target < len substring
mov dx, 0ffffh   ; Initialize dx to -1

@@10:
inc dx           ; For i = 0 to last possible index
mov cl, [byte bx + di] ; Save char at s[bx] in cl
mov [byte bx + di], ASCNull ; Replace char with null
call StrCompare  ; Compare si to altered di

```



## 2. Dizi Giriş/Çıkış Altyordamları

%TITLE "String Input/Output Routines"

IDEAL  
DOSSEG  
MODEL small

;--- Equates

BufSize EQU 255 ; Maximum string size (<=255)  
ASCnull EQU 0 ; ASCII null  
ASCCr EQU 13 ; ASCII carriage return  
ASCLf EQU 10 ; ASCII line feed

;--- String buffer structure for DOS function 0Ah

STRUC StrBuffer  
maxlen db BufSize ; Maximum buffer length  
strlen db 0 ; String length  
chars db BufSize DUP (?) ; Buffer for StrRead  
ENDS StrBuffer

DATASEG

buffer StrBuffer <> ; Buffer variable for ReadStr

CODESEG

;--- from: STRING.OBJ

EXTRN StrLength:proc, StrCopy:proc

PUBLIC StrRead, StrWrite, StrWrite2, NewLine

%NEWPAGE

---

; StrRead Read string with editing keys

---

; Input:

; di = address of destination string  
; cl = maximum string length EXCLUDING null terminator  
; Note: if cl = 0, StrRead does nothing

```
; Note: actual variable must be cl + 1 bytes long
; Note: string length is limited to 255 characters
; Output:
;   String copied from standart input into your buffer
; Registers:
;   none
```

---

```
PROC      StrRead
    or     cl, cl                ; Is cl = 0?
    jz     @@99                 ; If yes, jump to exit

    push   ax                   ; Save modified registers
    push   bx
    push   dx
    push   si

    mov     [buffer.maxlen], cl    ; Set maxlen byte
    mov     ah, 0Ah               ; DOS Buffered-Input function
    mov     dx, OFFSET buffer.maxlen ; Address struc with ds:dx
    int     21h                  ; Call DOS to read string
    xor     bh, bh                ; Zero high byte of bx
    mov     bl, [buffer.strlen]    ; bx = # chars in buffer
    mov     [bx + buffer.chars], ASCNull ; Change cr to null
    mov     si, OFFSET buffer.chars ; Address buffer with si
    call    StrCopy               ; Copy chars to user string

    pop     si                   ; Restore registers
    pop     dx
    pop     bx
    pop     ax

@@99:
    ret                          ; Return to caller
ENDP      StrRead
%NEWPAGE
```

---

```
; StrWrite/StrWrite2      Write string to standart output
```

---

```
; Input:
;   di = address of string (s)
;   cx = number of chars to write (StrWrite2 only)
; Output:
;   string s copied to standart output
; Registers:
;   cx (StrWrite only)
```

---

```
PROC      StrWrite
    call    StrLength            ; Set cx = length of string
```

```

PROC      StrWrite2                ; Alternate entry point
        push ax                    ; Save modified registers
        push bx
        push dx

        mov bx, 1                  ; Standart output handle
        mov dx, di                 ; ds:dx address string
        mov ah, 40h               ; DOS write to file or device
        int 21h                   ; Call DOS (on ret ax = # chars written)

        pop dx                    ; Restore registers
        pop bx
        pop ax
        ret                       ; Return to caller
ENDP      StrWrite2                ; End of alternate procedure
ENDP      StrWrite                 ; End of normal procedure

```

%NEWPAGE

```

;-----
; NewLine          Start new line on standart output file
;-----
; Input:
;   none
; Output:
;   carriage return, line feed sent to standart output
; Registers:
;   ah, dl
;-----
PROC      NewLine
        mov ah, 2                  ; DOS write-char routine
        mov dl, ASCCr             ; Load carriage return into dl
        int 21h                   ; Write carriage return
        mov dl, ASCLf             ; Load line feed into dl
        int 21h                   ; Write line feed
        ret                       ; Return to caller
ENDP      NewLine

        END                       ; End of STRIO module

```

## EK-2

**Dosya Yaratma, Anahtar ve Ana Dosyaya Kayıt Yapılmasında ve  
Dosyaların Okunmasında Kullanılan Program**

%TITLE "Ön Belleğe Bilgi Yükleme, Dosya açma, yazma, okumayı test  
30.1.1997 Tekirdağ"

```
IDEAL
DOSSEG
MODEL      small
STACK      256
```

```
MaxLen      EQU 65h          ; 100-character string
cr           EQU 13           ; ASCII carriage return
lf           EQU 10           ; ASCII line feed
filemax      EQU 0Dh          ; Filename 12 character
keymax       EQU 0Bh          ; key 10 character
keybuf       EQU 0Fh          ; kaydedilecek bilgi 14 byte
keybuf1      EQU 0Bh
big          EQU 8Dh          ;141 Byte KEY.DAT'tan bir defada okunacak bilgi
T            EQU 0Fh          ; KEY.DAT'tan oku.kayıt uzunluğu
can0         EQU 0Ch
```

```
DATASEG
```

```
exitCode     db 0
can          db 0Dh,09h,09h,09h,09h,09h,'      Gürcan UZAL',cr,lf,lf,0
baslik       db 0Dh,09h,09h,09h,'=====MENU=====',cr,lf,0
welcome      db 0Dh,09h,09h,09h,'*****',cr,lf
             db 0Dh,09h,09h,09h,'* Dosya Yaratma..... :1      *',cr,lf
             db 0Dh,09h,09h,09h,'* Dosyaya Yazma..... : 2      *',cr,lf
             db 0Dh,09h,09h,09h,'* Dosyayı Okuma.....: 3      *',cr,lf
             db 0Dh,09h,09h,09h,'*                               *',cr,lf
             db 0Dh,09h,09h,09h,'*      Çıkış..... : 4      *',cr,lf
             db 0Dh,09h,09h,09h,'*****',cr,lf,lf,0
secenek      db 0Dh,09h,09h,09h,'      SEÇENEĞİNİZ 1 - 4 : ',0
m1           db 0Dh,09h,09h,09h,' Dosya İsmi = ',0
m2           db 0Dh,09h,09h,09h,' 1 adet dosya yaratıldı.....',0
m3           db 0Dh,09h,09h,09h,' Dosya yaratılamadı.....!',0
m4           db 0Dh,09h,09h,09h,' Dosya açılamadı.....!',0
myaz         db 0Dh,09h,09h,09h,' Kayıt yapılacak dosya ismi : ',0
moku         db 0Dh,09h,09h,09h,' Kayıt okunulacak dosya ismi : ',0
mkayityok    db 0Dh,09h,09h,09h,' Bu isimde bir kayıt yok.....!',0
mesaj1       db 0Dh,'***** ile başlayan başka kayıt varmı.... ? (E/H):',0
```

|             |    |                                                                                   |
|-------------|----|-----------------------------------------------------------------------------------|
| yeryok      | db | 0Dh,09h,' İlk iki karakteri aynı olan anahtarların kaydedildiği yer doldu....!',0 |
| kayit       | db | 0Dh,09h,09h,09h,' Başka Kayıt Girişi varmı...? (E/H):',0                          |
| okuma       | db | 0Dh,09h,09h,09h,' Başka Okunacak Kayıt varmı...? (E/H):',0                        |
| mnotfound   | db | 0Dh,09h,09h,09h,' Aranılan anahtar bulunamadı....!',0                             |
| don         | db | 0Dh,09h,09h,09h,' Ana menü için ENTER....',0                                      |
| bilgi       | db | 0Dh,09h,' Kayıt edilecek bilgiyi girin: ',0                                       |
| mara        | db | 0Dh,09h,09h,09h,' Aranılacak alt dizi = ',0                                       |
| dgoster1    | db | 0Dh,09h,09h,09h,' KEY.DAT in file pointeri ayarlanmadı..!',0                      |
| dgoster2    | db | 0Dh,09h,' MASTER.DAT in file pointeri ayarlanmadı..!',0                           |
| acilmadi    | db | 0Dh,09,' MASTER.DAT dosyası açılmadı.....!',0                                     |
| testString  | db | MaxLen DUP (?) ; MaxLen chars + null                                              |
| ;FileName   | db | filemax DUP (?) ; Ana Dosya ismi                                                  |
| KeyFileName | db | 'KEY.DAT',0 ; Anahtar Dosya ismi                                                  |
| FileName    | db | 'MASTER.DAT',0                                                                    |
| handle      | dw | ?                                                                                 |
| handle1     | dw | ?                                                                                 |
| say         | dw | ?                                                                                 |
| key         | db | keybuf DUP (?) ; Aranılacak alt dizi                                              |
| key1        | db | keybuf1 DUP (?)                                                                   |
| BigBuffer   | db | big DUP (?) ; Key'in aranılacağı buffer                                           |
| target      | db | T DUP (0), 0 ; KEY.DAT'tan okunacak kayıt alanı                                   |
| harf1       | db | 1 DUP (0), 0 ; 1.karakter alanı                                                   |
| harf2       | db | 1 DUP (0), 0 ; 2.karakter alanı                                                   |
| harf3       | db | 1 DUP (0), 0 ; 3.karakter alanı                                                   |
| harf4       | db | 1 DUP (0), 0 ; 4.karakter alanı                                                   |
| harf5       | db | 1 DUP (0), 0 ; 5.karakter alanı                                                   |
| mkey        | db | 0Dh,09h,' Kayıt için anahtar kelimeyi yazınız: ',0                                |
| mkey1       | db | 0Dh,09h,' Okuma için anahtar kelimeyi yazınız: ',0                                |
| G           | dw | ? ; Anahtarın 1.karakterinin sayısal değ. sak. alan                               |
| H           | dw | ? ; Anahtarın 2.karakterinin sayısal değ. sak. alan                               |
| I           | dw | ? ; Anahtarın 3.karakterinin sayısal değ. sak. alan                               |
| J           | dw | ? ; Anahtarın 4.karakterinin sayısal değ. sak. alan                               |
| C           | dw | 5F45h ;                                                                           |
| K           | dw | 2DFEh ;                                                                           |
| W           | dw | 0349h ;                                                                           |
| RecSize     | dw | 0065h ; Ana dosya için kayıt uzn.100 byte (64+Null=65h)                           |
| AA0X        | dw | ? ; 1.çarpımın düşük sözcüğünün saklandığı alan                                   |
| BB0X        | dw | ? ; yüksek sözcüğünün saklandığı alan                                             |
| AA1X        | dw | ? ; 2.çarpımın düşük sözcüğünün saklandığı alan                                   |
| BB1X        | dw | ? ; yüksek sözcüğünün saklandığı alan                                             |
| AA2X        | dw | ? ; 3.çarpımın düşük sözcüğünün saklandığı alan                                   |
| BB2X        | dw | ? ; yüksek sözcüğünün saklandığı alan                                             |
| AA3X        | dw | ? ; 4.çarpımın düşük sözcüğünün saklandığı alan                                   |

|         |    |                 |                                                     |
|---------|----|-----------------|-----------------------------------------------------|
| BB3X    | dw | ?               | ; yüksek sozcüğünün saklandığı alan                 |
| AA4X    | dw | ?               | ; 5.çarpımın düşük sozcüğünün saklandığı alan       |
| BB4X    | dw | ?               | ; yüksek sozcüğünün saklandığı alan                 |
| AA11X   | dw | ?               | ;                                                   |
| BB11X   | dw | ?               | ;                                                   |
| AA22X   | dw | ?               | ;                                                   |
| BB22X   | dw | ?               | ;                                                   |
| AA33X   | dw | ?               | ;                                                   |
| BB33X   | dw | ?               | ;                                                   |
| AA44X   | dw | ?               | ;                                                   |
| BB44X   | dw | ?               | ;                                                   |
| TOP0    | dw | ?               | ; Çarpımların düşük sözcüklerinin top. sak.alan     |
| TOP1    | dw | ?               | ; Çarpımların yüksek sözcüklerinin top.sak.alan     |
| ADRESL  | dw | ?               | ; Anahtara karş. gelen düşük adresin sak. alan      |
| ADRESH  | dw | ?               | ; Anahtara karş. gelen yüksek adresin sak alan      |
| MADRESL | dw | ?               | ; MASTER.DAT'ın düşük adr.sak.alan                  |
| MADRESH | dw | ?               | ; MASTER.DAT'ın yüksek adr.sak.alan                 |
| MAXAREA | dw | ?               | ; Anahtar kelimenin aranılacağı alanın max.uzunluğu |
| saatbuf | db | can0 DUP (0), 0 |                                                     |
| bolen   | db | 0Ah             |                                                     |
| saat    | dw | ?               |                                                     |
| dakika  | dw | ?               |                                                     |
| saniye  | dw | ?               |                                                     |
| salise  | dw | ?               |                                                     |

## CODESEG

;----- From STRIO.OBJ:

EXTRN StrRead:proc, StrWrite:proc, NewLine:proc

;----- From STRING.OBJ:

EXTRN StrInsert:proc, StrPos:proc, StrNull:proc, StrLength:proc

;===== KARAKTERLERİ VE YANINDAKİ ADR. ÖNBELLEĞE YÜKLEME =====

Start: mov ax, @data

```

mov  ds, ax
mov  es, ax

```

```

push ds          ; Programın başlangıç
push es          ; adresini sakla

```

;===== Ön belleğe bilginin yüklenmesi -----

```

mov  ax, 0D000h   ; Bellek kartının yer. adresin
mov  es, ax       ; segment adresini
mov  di, 00000h   ; offset adresini yükle

```

;----- 1.harflerin yanına ikinci harflerin baş.adresinin yüklenmesi

```

mov  al,'A'
stosb
mov  ax,00057h
stosw

```

```

mov  al,'B'
stosb
mov  ax,00122h
stosw

```

```

mov  al,'C'
stosb
mov  ax,001EDh
stosw

```

```

mov  al,'Ç'
stosb
mov  ax,002B8h
stosw

```

```

mov  al,'D'
stosb
mov  ax,00383h
stosw

```

```

mov  al,'E'
stosb
mov  ax,0044Eh
stosw

```

```

mov  al,'F'
stosb
mov  ax,00519h
stosw

```

```
mov  al,'G'  
stosb  
mov  ax,005E4h  
stosw
```

```
mov  al,'Ě'  
stosb  
mov  ax,006AFh  
stosw
```

```
mov  al,'H'  
stosb  
mov  ax,0077Ah  
stosw
```

```
mov  al,'I'  
stosb  
mov  ax,00845h  
stosw
```

```
mov  al,'I'  
stosb  
mov  ax,00910h  
stosw
```

```
mov  al,'J'  
stosb  
mov  ax,009DBh  
stosw
```

```
mov  al,'K'  
stosb  
mov  ax,00AA6h  
stosw
```

```
mov  al,'L'  
stosb  
mov  ax,00B71h  
stosw
```

```
mov  al,'M'  
stosb  
mov  ax,00C3Ch  
stosw
```

```
mov  al,'N'  
stosb  
mov  ax,00D07h  
stosw
```

```
mov  al,'O'  
stosb  
mov  ax,00DD2h  
stosw
```

```
mov  al,'Ö'  
stosb  
mov  ax,00E9Dh  
stosw
```

```
mov  al,'P'  
stosb  
mov  ax,00F68h  
stosw
```

```
mov  al,'R'  
stosb  
mov  ax,01033h  
stosw
```

```
mov  al,'S'  
stosb  
mov  ax,010FEh  
stosw
```

```
mov  al,'Ş'  
stosb  
mov  ax,011C9h  
stosw
```

```
mov  al,'T'  
stosb  
mov  ax,01294h  
stosw
```

```
mov  al,'U'  
stosb  
mov  ax,0135Fh  
stosw
```

```
mov  al,'Ü'  
stosb  
mov  ax,0142Ah  
stosw
```

```
mov  al,'V'  
stosb  
mov  ax,014F5h
```

```
stosw
```

```
mov al,'Y'
stosb
mov ax,015C0h
stosw
```

```
mov al,'Z'
stosb
mov ax,0168Bh
stosw
```

;===== A\* ikinci harfin yanındaki bilginin yüklenmesi

```
mov al,'A'
stosb
mov ax,00000h
stosw
mov ax,0008Dh
stosw
mov ax,0008Dh
stosw
```

```
mov al,'B'
stosb
mov ax,00000h
stosw
mov ax,0011Ah
stosw
mov ax,0008Dh
stosw
```

```
mov al,'C'
stosb
mov ax,00000h
stosw
mov ax,001A7h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'Ç'
stosb
mov ax,00000h
stosw
mov ax,00234h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'D'
stosb
mov ax,00000h
stosw
mov ax,002C1h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'E'
stosb
mov ax,00000h
stosw
mov ax,0034Eh
stosw
mov ax,0008Dh
stosw
```

```
mov al,'F'
stosb
mov ax,00000h
stosw
mov ax,003DBh
stosw
mov ax,0008Dh
stosw
```

```
mov al,'G'
stosb
mov ax,00000h
stosw
mov ax,00468h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'Ě'
stosb
mov ax,00000h
stosw
mov ax,004F5h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'H'
stosb
mov ax,00000h
```

```
stosw
mov ax,00582h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'I'
stosb
mov ax,00000h
stosw
mov ax,0060Fh
stosw
mov ax,0008Dh
stosw
```

```
mov al,'I'
stosb
mov ax,00000h
stosw
mov ax,0069Ch
stosw
mov ax,0008Dh
stosw
```

```
mov al,'J'
stosb
mov ax,00000h
stosw
mov ax,00729h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'K'
stosb
mov ax,00000h
stosw
mov ax,007B6h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'L'
stosb
mov ax,00000h
stosw
mov ax,00843h
stosw
mov ax,0008Dh
```

stosw

```
mov  al,'M'
stosb
mov  ax,00000h
stosw
mov  ax,008D0h
stosw
mov  ax,0008Dh
stosw
```

```
mov  al,'N'
stosb
mov  ax,00000h
stosw
mov  ax,0095Dh
stosw
mov  ax,0008Dh
stosw
```

```
mov  al,'O'
stosb
mov  ax,00000h
stosw
mov  ax,009EAh
stosw
mov  ax,0008Dh
stosw
```

```
mov  al,'Ö'
stosb
mov  ax,00000h
stosw
mov  ax,00A77h
stosw
mov  ax,0008Dh
stosw
```

```
mov  al,'P'
stosb
mov  ax,00000h
stosw
mov  ax,00B04h
stosw
mov  ax,0008Dh
stosw
```

```
mov  al,'R'
stosb
```

```
mov ax,00000h
stosw
mov ax,00B91h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'S'
stosb
mov ax,00000h
stosw
mov ax,00C1Eh
stosw
mov ax,0008Dh
stosw
```

```
mov al,'$'
stosb
mov ax,00000h
stosw
mov ax,00CABh
stosw
mov ax,0008Dh
stosw
```

```
mov al,'T'
stosb
mov ax,00000h
stosw
mov ax,00D38h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'U'
stosb
mov ax,00000h
stosw
mov ax,00DC5h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'Ü'
stosb
mov ax,00000h
stosw
mov ax,00E52h
stosw
```

```
mov ax,0008Dh
stosw
```

```
mov al,'V'
stosb
mov ax,00000h
stosw
mov ax,00EDFh
stosw
mov ax,0008Dh
stosw
```

```
mov al,'Y'
stosb
mov ax,00000h
stosw
mov ax,00F6Ch
stosw
mov ax,0008Dh
stosw
```

```
mov al,'Z'
stosb
mov ax,00000h
stosw
mov ax,00FF9h
stosw
mov ax,0008Dh
stosw
```

;===== B\* ikinci harfin yanındaki bilginin yüklenmesi

```
mov al,'A'
stosb
mov ax,00000h
stosw
mov ax,01086h
stosw
mov ax,0008Dh
stosw
```

```
.
.
.
.
.
.
```

;===== Z\* ikinci harfin yanındaki bilginin yüklenmesi

```
mov    al,'A'
stosb
mov    ax,00001h
stosw
mov    ax,0BFC9h
stosw
mov    ax,0008Dh
stosw
```

```
mov    al,'B'
stosb
mov    ax,00001h
stosw
mov    ax,0C056h
stosw
mov    ax,0008Dh
stosw
```

```
mov    al,'C'
stosb
mov    ax,00001h
stosw
mov    ax,0C0E3h
stosw
mov    ax,0008Dh
stosw
```

```
mov    al,'Ç'
stosb
mov    ax,00001h
stosw
mov    ax,0C170h
stosw
mov    ax,0008Dh
stosw
```

```
mov    al,'D'
stosb
mov    ax,00001h
stosw
mov    ax,0C1FDh
stosw
mov    ax,0008Dh
stosw
```

```
mov    al,'E'
stosb
```

```
mov ax,00001h
stosw
mov ax,0C28Ah
stosw
mov ax,0008Dh
stosw
```

```
mov al,'F'
stosb
mov ax,00001h
stosw
mov ax,0C317h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'G'
stosb
mov ax,00001h
stosw
mov ax,0C3A4h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'Ĝ'
stosb
mov ax,00001h
stosw
mov ax,0C431h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'H'
stosb
mov ax,00001h
stosw
mov ax,0C4BEh
stosw
mov ax,0008Dh
stosw
```

```
mov al,'I'
stosb
mov ax,00001h
stosw
mov ax,0C54Bh
stosw
```

```
mov ax,0008Dh
stosw
```

```
mov al,'I'
stosb
mov ax,00001h
stosw
mov ax,0C5D8h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'J'
stosb
mov ax,00001h
stosw
mov ax,0C665h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'K'
stosb
mov ax,00001h
stosw
mov ax,0C6F2h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'L'
stosb
mov ax,00001h
stosw
mov ax,0C77Fh
stosw
mov ax,0008Dh
stosw
```

```
mov al,'M'
stosb
mov ax,00001h
stosw
mov ax,0C80Ch
stosw
mov ax,0008Dh
stosw
```

```
mov al,'N'
```

```
stosb
mov  ax,00001h
stosw
mov  ax,0C899h
stosw
mov  ax,0008Dh
stosw
```

```
mov  al,'O'
stosb
mov  ax,00001h
stosw
mov  ax,0C926h
stosw
mov  ax,0008Dh
stosw
```

```
mov  al,'Ö'
stosb
mov  ax,00001h
stosw
mov  ax,0C9B3h
stosw
mov  ax,0008Dh
stosw
```

```
mov  al,'P'
stosb
mov  ax,00001h
stosw
mov  ax,0CA40h
stosw
mov  ax,0008Dh
stosw
```

```
mov  al,'R'
stosb
mov  ax,00001h
stosw
mov  ax,0CACDh
stosw
mov  ax,0008Dh
stosw
```

```
mov  al,'S'
stosb
mov  ax,00001h
stosw
mov  ax,0CB5Ah
```

```
stosw
mov ax,0008Dh
stosw
```

```
mov al,'$'
stosb
mov ax,00001h
stosw
mov ax,0CBE7h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'T'
stosb
mov ax,00001h
stosw
mov ax,0CC74h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'U'
stosb
mov ax,00001h
stosw
mov ax,0CD01h
stosw
mov ax,0008Dh
stosw
```

```
mov al,'Ü'
stosb
mov ax,00001h
stosw
mov ax,0CD8Eh
stosw
mov ax,0008Dh
stosw
```

```
mov al,'V'
stosb
mov ax,00001h
stosw
mov ax,0CE1Bh
stosw
mov ax,0008Dh
stosw
```

```

mov    al,'Y'
stosb
mov    ax,00001h
stosw
mov    ax,0CEA8h
stosw
mov    ax,0008Dh
stosw

```

```

mov    al,'Z'
stosb
mov    ax,00001h
stosw
mov    ax,0CF35h
stosw
mov    ax,0008Dh
stosw

```

```

pop    es
pop    ds

```

----- Ekranı Silme -----

```

Start1:  mov    cx, 0000h    ;sadır,sütün sol üst köşe
          mov    dx, 2479h   ;sadır,sütün sağ alt köşe
          mov    bh, 07      ;normal özellik
          mov    ah, 06      ;aktif pencereyi yukarı kaydırma
          mov    al, 00       ;ekranı silme
          int     10h         ; EKRANI TEMİZLEME

```

----- Menüü Ekrana Yazma -----

```

mov    di, OFFSET can
call   StrWrite
mov    di, OFFSET baslik
call   StrWrite
mov    di, OFFSET welcome    ; Display welcome message
call   StrWrite
mov    di, OFFSET secenek
call   StrWrite

```

----- Menüden Seçeneğin Alınması -----

```

mov    ah, 01h    ; klavyeden karakter oku
int     21h        ; ekrana yaz
cmp    al, 49      ; Seçenek 1 (ASCII 49 = 1) Dosya Yaratma
je     yarat
jmp    x0
x0:    cmp    al, 50    ; Seçenek 2 (ASCII 50 = 2) Dosyaya Yazma

```

```

        je      yaz1
        jmp     a0
yaz1:   jmp     yaz
a0:     cmp     al, 51          ; Seçenek 3 (ASCII 51 = 3) Dosyayı Okuma
        je      oku1
        jmp     b0
oku1:   jmp     oku
b0:     cmp     al, 52          ; Seçenek 4 (ASCII 52 = 4) Çıkış
        je      Exit1
Exit1:  jmp     Exit

```

===== Dosya Yaratmak =====

```

yarat:   call    NewLine
        mov     di, OFFSET m1      ;Ekрана'Dosya ismi ='yaz
        call    StrWrite
        mov     di, OFFSET FileName
        mov     cx, filemax        ; Dosya ismi 14 karakter
        call    StrRead           ; Dosya ismini oku
        mov     ah, 3Ch           ; DOS dosya yaratma fonksiyonu
        mov     dx, OFFSET FileName
        mov     cx, 00h           ; Normal dosya açma
        int     21h
        jc      error
        mov     [handle], ax
        call    NewLine
        mov     di, OFFSET m2      ; '1 adet dosya yaratıldı...'mesajı
        call    StrWrite
        jmp     cik
error:   call    NewLine
        mov     di, OFFSET m3      ; 'Dosya yaratılamadı....!'mesajı
        call    StrWrite
cik:     call    NewLine
        mov     di, OFFSET don      ; 'Ana menü için ENTER...'mesajı
        call    StrWrite
        mov     ah, 01h           ; Ana menüye dönüş
        int     21h
        jmp     Start1

```

===== Dosyaya Bilgi Yazma =====

----- Yazma için KEY.DAT ve MASTER.DAT dosyalarının açılması-----

```

yaz:     mov     ah, 3Dh           ;KEY.DAT dosyasını
        mov     al, 01            ;yalnızca yazma
        mov     dx, OFFSET KeyFileName ;için aç
        int     21h

```

```

mov [handle], ax

mov ah, 3Dh           ;MASTER.DAT dosyasını
mov al, 01           ;yalnızca yazma
mov dx, OFFSET FileName ;için aç
int 21h
mov [handle1], ax

wwwwwwr: mov dx, 00000h
push dx

wwwwww:  mov di, OFFSET key
call StrNull

call NewLine          ; 'Kayıt edilecek bilgiyi girin'
mov di, OFFSET bilgi  ; mesajını ekrana
call StrWrite         ; yaz

;----- Kayıt Yapılacak Bilgiyi Alma -----

mov di, OFFSET testString
call StrNull

mov di, OFFSET testString ; di = testString'in adresi
mov cx, MaxLen            ; cx = max. genişlik (100 bytes)
call StrRead              ; Klavyeden diziyi alma

;----- Kayıt için anahtar kelimenin alınması -----

call NewLine
mov di, OFFSET mkey
call StrWrite

mov di, OFFSET key          ; Anahtar kelimenin
mov cx, keymax              ; 10 bayt
call StrRead                ; klavyeden alınması

;----- Anahtar kelimenin ilk üç karakterinin karşılaştırma için alınması

mov si, OFFSET key          ; Anahtarın 1.karakterini
lods [key]                  ; AL'ye yükle
mov di, OFFSET harf1        ; AL'nin içeriğinin
mov [harf1], al             ; harf1'e yüklenmesi

cld                          ; si<--si+1, di<--di+1

lods [key]                  ; Anahtarın 2.karakterini AL'ye yükle

```

```

mov  di, OFFSET harf2 ; AL'nin içeriğinin
mov  [harf2], al      ; harf2'ye yüklenmesi

cld

lods  [key]           ; Anahtarın 3.karakterini AL'ye yükle
mov  di, OFFSET harf3 ; AL'nin içeriğinin
mov  [harf3], al      ; harf3'e yüklenmesi

```

;----- Anahtarın 1.harfinin Sayı Karşılığının Tesbiti--

```

mov  si, OFFSET harf1 ; Anahtarın 1.harfinin
lods  [harf1]          ; bl'ye yüklenmesi
mov  bl, al

A22:  cmp  bl, 'A'
      jne  B22
      mov  ax, 00h
      mov  [G], ax      ; G = 1 , G = G - 1 , G = 0
      jmp  k3           ; Anahtarın 2.karakterinin sayısal tespitine atla

B22:  cmp  bl, 'B'
      jne  C22
      mov  ax, 01h
      mov  [G], ax      ; G = 2 , G = G - 1 , G = 1
      jmp  k3

C22:  cmp  bl, 'C'
      jne  CC22
      mov  ax, 02h
      mov  [G], ax      ; G = 3 , G = G - 1 , G = 2
      jmp  k3

CC22: cmp  bl, 'Ç'
      jne  D22
      mov  ax, 03h
      mov  [G], ax      ; G = 4 , G = G - 1 , G = 3
      jmp  k3

D22:  cmp  bl, 'D'
      jne  E22
      mov  ax, 04h
      mov  [G], ax      ; G = 5 , G = G - 1 , G = 4
      jmp  k3

E22:  cmp  bl, 'E'
      jne  F22

```

```

        mov ax, 05h
        mov [G], ax          ; G = 6 , G = G - 1 , G = 5
        jmp k3

F22:    cmp bl, 'F'
        jne G22
        mov ax, 06h
        mov [G], ax          ; G = 7 , G = G - 1 , G = 6
        jmp k3

G22:    cmp bl, 'G'
        jne GG22
        mov ax, 07h
        mov [G], ax          ; G = 8 , G = G - 1 , G = 7
        jmp k3

GG22:   cmp bl, 'Ĝ'
        jne H22
        mov ax, 08h
        mov [G], ax          ; G = 9 , G = G - 1 , G = 8
        jmp k3

H22:    cmp bl, 'H'
        jne I22
        mov ax, 09h
        mov [G], ax          ; G = A , G = G - 1 , G = 9
        jmp k3

I22:    cmp bl, 'I'
        jne II22
        mov ax, 0Ah
        mov [G], ax          ; G = B , G = G - 1 , G = A
        jmp k3

II22:   cmp bl, 'İ'
        jne J22
        mov ax, 0Bh
        mov [G], ax          ; G = C , G = G - 1 , G = B
        jmp k3

J22:    cmp bl, 'J'
        jne K22
        mov ax, 0Ch
        mov [G], ax          ; G = D , G = G - 1 , G = C
        jmp k3

K22:    cmp bl, 'K'
        jne L22
        mov ax, 0Dh

```

```

        mov    [G], ax          ; G = E , G = G - 1 , G = D
        jmp    k3

L22:    cmp    bl, 'L'
        jne    M22
        mov    ax, 0Eh
        mov    [G], ax          ; G = F , G = G - 1 , G = E
        jmp    k3

M22:    cmp    bl, 'M'
        jne    N22
        mov    ax, 0Fh
        mov    [G], ax          ; G = 10 , G = G - 1 , G = F
        jmp    k3

N22:,   cmp    bl, 'N'
        jne    O22
        mov    ax, 10h
        mov    [G], ax          ; G = 11 , G = G - 1 , G = 10
        jmp    k3

O22:    cmp    bl, 'O'
        jne    OO22
        mov    ax, 11h
        mov    [G], ax          ; G = 12 , G = G - 1 , G = 11
        jmp    k3

OO22:   cmp    bl, 'Ö'
        jne    P22
        mov    ax, 12h
        mov    [G], ax          ; G = 13 , G = G - 1 , G = 12
        jmp    k3

P22:    cmp    bl, 'P'
        jne    R22
        mov    ax, 13h
        mov    [G], ax          ; G = 14 , G = G - 1 , G = 13
        jmp    k3

R22:    cmp    bl, 'R'
        jne    S22
        mov    ax, 14h
        mov    [G], ax          ; G = 15 , G = G - 1 , G = 14
        jmp    k3

S22:    cmp    bl, 'S'
        jne    SS22
        mov    ax, 15h
        mov    [G], ax          ; G = 16 , G = G - 1 , G = 15

```

```

        jmp    k3

SS22:   cmp    bl, '$'
        jne    T22
        mov    ax, 16h
        mov    [G], ax          ; G = 17 , G = G - 1 , G = 16
        jmp    k3

T22:    cmp    bl, 'T'
        jne    U22
        mov    ax, 17h
        mov    [G], ax          ; G = 18 , G = G - 1 , G = 17
        jmp    k3

U22:    cmp    bl, 'U'
        jne    UU22
        mov    ax, 18h
        mov    [G], ax          ; G = 19 , G = G - 1 , G = 18
        jmp    k3

UU22:   cmp    bl, 'Ü'
        jne    V22
        mov    ax, 19h
        mov    [G], ax          ; G = 1A , G = G - 1 , G = 19
        jmp    k3

V22:    cmp    bl, 'V'
        jne    Y22
        mov    ax, 1Ah
        mov    [G], ax          ; G = 1B , G = G - 1 , G = 1A
        jmp    k3

Y22:    cmp    bl, 'Y'
        jne    Z22
        mov    ax, 1Bh
        mov    [G], ax          ; G = 1C , G = G - 1 , G = 1B
        jmp    k3

Z22:    cmp    bl, 'Z'
        mov    ax, 1Ch
        mov    [G], ax          ; G = 1D , G = G - 1 , G = 1C

```

k3: ;----- Anahtarın 2.karakterinin Sayı Karşılığının Tesbiti--

```

        mov    si, OFFSET harf2 ; Anahtarın 2.harfinin
        lods    [harf2]          ; bl'ye yüklenmesi
        mov    bl, al

```

```

A33:    cmp    bl, 'A'
        jne    B33
        mov    ax, 00h
        mov    [H], ax          ; H = 1 , H = H - 1 , H = 0
        jmp    k4              ; Anahtarın 3.karakterinin sayısal tesbitine atla

B33:    cmp    bl, 'B'
        jne    C33
        mov    ax, 01h
        mov    [H], ax          ; H = 2 , H = H - 1 , H = 1
        jmp    k4

C33:    cmp    bl, 'C'
        jne    CC33
        mov    ax, 02h
        mov    [H], ax          ; H = 3 , H = H - 1 , H = 2
        jmp    k4

CC33:   cmp    bl, 'Ç'
        jne    D33
        mov    ax, 03h
        mov    [H], ax          ; H = 4 , H = H - 1 , H = 3
        jmp    k4

D33:    cmp    bl, 'D'
        jne    E33
        mov    ax, 04h
        mov    [H], ax          ; H = 5 , H = H - 1 , H = 4
        jmp    k4

E33:    cmp    bl, 'E'
        jne    F33
        mov    ax, 05h
        mov    [H], ax          ; H = 6 , H = H - 1 , H = 5
        jmp    k4

F33:    cmp    bl, 'F'
        jne    G33
        mov    ax, 06h
        mov    [H], ax          ; H = 7 , H = H - 1 , H = 6
        jmp    k4

G33:    cmp    bl, 'G'
        jne    GG33
        mov    ax, 07h
        mov    [H], ax          ; H = 8 , H = H - 1 , H = 7
        jmp    k4

GG33:   cmp    bl, 'Ğ'

```

```

    jne    H33
    mov    ax, 08h
    mov    [H], ax          ; H = 9 , H = H - 1 , H = 8
    jmp    k4

H33:    cmp    bl, 'H'
    jne    I33
    mov    ax, 09h
    mov    [H], ax          ; H = A , H = H - 1 , H = 9
    jmp    k4

I33:    cmp    bl, 'I'
    jne    II33
    mov    ax, 0Ah
    mov    [H], ax          ; H = B , H = H - 1 , H = A
    jmp    k4

II33:   cmp    bl, 'I'
    jne    J33
    mov    ax, 0Bh
    mov    [H], ax          ; H = C , H = H - 1 , H = B
    jmp    k4

J33:    cmp    bl, 'J'
    jne    K33
    mov    ax, 0Ch
    mov    [H], ax          ; H = D , H = H - 1 , H = C
    jmp    k4

K33:    cmp    bl, 'K'
    jne    L33
    mov    ax, 0Dh
    mov    [H], ax          ; H = E , H = H - 1 , H = D
    jmp    k4

L33:    cmp    bl, 'L'
    jne    M33
    mov    ax, 0Eh
    mov    [H], ax          ; H = F , H = H - 1 , H = E
    jmp    k4

M33:    cmp    bl, 'M'
    jne    N33
    mov    ax, 0Fh
    mov    [H], ax          ; H = 10 , H = H - 1 , H = F
    jmp    k4

N33:    cmp    bl, 'N'
    jne    O33

```

```

        mov ax, 10h
        mov [H], ax          ; H = 11 , H = H - 1 , H = 10
        jmp k4

O33:    cmp bl, 'O'
        jne OO33
        mov ax, 11h
        mov [H], ax          ; H = 12 , H = H - 1 , H = 11
        jmp k4

OO33:   cmp bl, 'Ö'
        jne P33
        mov ax, 12h
        mov [H], ax          ; H = 13 , H = H - 1 , H = 12
        jmp k4

P33:    cmp bl, 'P'
        jne R33
        mov ax, 13h
        mov [H], ax          ; H = 14 , H = H - 1 , H = 13
        jmp k4

R33:    cmp bl, 'R'
        jne S33
        mov ax, 14h
        mov [H], ax          ; H = 15 , H = H - 1 , H = 14
        jmp k4

S33:    cmp bl, 'S'
        jne SS33
        mov ax, 15h
        mov [H], ax          ; H = 16 , H = H - 1 , H = 15
        jmp k4

SS33:   cmp bl, 'Ş'
        jne T33
        mov ax, 16h
        mov [H], ax          ; H = 17 , H = H - 1 , H = 16
        jmp k4

T33:    cmp bl, 'T'
        jne U33
        mov ax, 17h
        mov [H], ax          ; H = 18 , H = H - 1 , H = 17
        jmp k4

U33:    cmp bl, 'U'
        jne UU33
        mov ax, 18h

```

```

        mov [H], ax          ; H = 19 , H = H - 1 , H = 18
        jmp k4

UU33:   cmp bl, 'Ü'
        jne V33
        mov ax, 19h
        mov [H], ax          ; H = 1A , H = H - 1 , H = 19
        jmp k4

V33:    cmp bl, 'V'
        jne Y33
        mov ax, 1Ah
        mov [H], ax          ; H = 1B , H = H - 1 , H = 1A
        jmp k4

Y33:    cmp bl, 'Y'
        jne Z33
        mov ax, 1Bh
        mov [H], ax          ; H = 1C , H = H - 1 , H = 1B
        jmp k4

Z33:    cmp bl, 'Z'
        mov ax, 1Ch
        mov [H], ax          ; H = 1D , H = H - 1 , H = 1C

k4: ; ————— Anahtarın 3.karakterinin Sayı Karşılığının Tesbiti—

        mov si, OFFSET harf3 ; Anahtarın 3.harfinin
        lods [harf3]          ; bl'ye yüklenmesi
        mov bl, al

A44:    cmp bl, 'A'
        jne B44
        mov ax, 00h
        mov [I], ax          ; I = 1 , I = I - 1 , I = 0
        jmp k5 ; Anahtarın 4.karakterinin sayısal tesbitine atla

B44:    cmp bl, 'B'
        jne C44
        mov ax, 01h
        mov [I], ax          ; I = 2 , I = I - 1 , I = 1
        jmp k5

C44:    cmp bl, 'C'
        jne CC44
        mov ax, 02h
        mov [I], ax          ; I = 3 , I = I - 1 , I = 2
        jmp k5

```

```

CC44:    cmp    bl, 'C'
         jne    D44
         mov    ax, 03h
         mov    [I], ax           ; I = 4 , I = I - 1 , I = 3
         jmp    k5

D44:     cmp    bl, 'D'
         jne    E44
         mov    ax, 04h
         mov    [I], ax           ; I = 5 , I = I - 1 , I = 4
         jmp    k5

E44:     cmp    bl, 'E'
         jne    F44
         mov    ax, 05h
         mov    [I], ax           ; I = 6 , I = I - 1 , I = 5
         jmp    k5

F44:     cmp    bl, 'F'
         jne    G44
         mov    ax, 06h
         mov    [I], ax           ; I = 7 , I = I - 1 , I = 6
         jmp    k5

G44:     cmp    bl, 'G'
         jne    GG44
         mov    ax, 07h
         mov    [I], ax           ; I = 8 , I = I - 1 , I = 7
         jmp    k5

GG44:    cmp    bl, 'Ĝ'
         jne    H44
         mov    ax, 08h
         mov    [I], ax           ; I = 9 , I = I - 1 , I = 8
         jmp    k5

H44:     cmp    bl, 'H'
         jne    I44
         mov    ax, 09h
         mov    [I], ax           ; I = A , I = I - 1 , I = 9
         jmp    k5

I44:     cmp    bl, 'I'
         jne    II44
         mov    ax, 0Ah
         mov    [I], ax           ; I = B , I = I - 1 , I = A
         jmp    k5

```

```

II44:    cmp    bl, 'I'
         jne    J44
         mov    ax, 0Bh
         mov    [I], ax           ; I = C , I = I - 1 , I = B
         jmp    k5

J44:     cmp    bl, 'J'
         jne    K44
         mov    ax, 0Ch
         mov    [I], ax          ; I = D , I = I - 1 , I = C
         jmp    k5

K44:     cmp    bl, 'K'
         jne    L44
         mov    ax, 0Dh
         mov    [I], ax          ; I = E , I = I - 1 , I = D
         jmp    k5

L44:     cmp    bl, 'L'
         jne    M44
         mov    ax, 0Eh
         mov    [I], ax          ; I = F , I = I - 1 , I = E
         jmp    k5

M44:     cmp    bl, 'M'
         jne    N44
         mov    ax, 0Fh
         mov    [I], ax          ; I = 10 , I = I - 1 , I = F
         jmp    k5

N44:     cmp    bl, 'N'
         jne    O44
         mov    ax, 10h
         mov    [I], ax          ; I = 11 , I = I - 1 , I = 10
         jmp    k5

O44:     cmp    bl, 'O'
         jne    OO44
         mov    ax, 11h
         mov    [I], ax          ; I = 12 , I = I - 1 , I = 11
         jmp    k5

OO44:    cmp    bl, 'Ö'
         jne    P44
         mov    ax, 12h
         mov    [I], ax          ; I = 13 , I = I - 1 , I = 12
         jmp    k5

P44:     cmp    bl, 'P'

```

```

jne    R44
mov    ax, 13h
mov    [I], ax          ; I = 14 , I = I - 1 , I = 13
jmp    k5

R44:   cmp    bl, 'R'
jne    S44
mov    ax, 14h
mov    [I], ax          ; I = 15 , I = I - 1 , I = 14
jmp    k5

S44:   cmp    bl, 'S'
jne    SS44
mov    ax, 15h
mov    [I], ax          ; I = 16 , I = I - 1 , I = 15
jmp    k5

SS44:  cmp    bl, 'Ş'
jne    T44
mov    ax, 16h
mov    [I], ax          ; I = 17 , I = I - 1 , I = 16
jmp    k5

T44:   cmp    bl, 'T'
jne    U44
mov    ax, 17h
mov    [I], ax          ; I = 18 , I = I - 1 , I = 17
jmp    k5

U44:   cmp    bl, 'U'
jne    UU44
mov    ax, 18h
mov    [I], ax          ; I = 19 , I = I - 1 , I = 18
jmp    k5

UU44:  cmp    bl, 'Ü'
jne    V44
mov    ax, 19h
mov    [I], ax          ; I = 1A , I = I - 1 , I = 19
jmp    k5

V44:   cmp    bl, 'V'
jne    Y44
mov    ax, 1Ah
mov    [I], ax          ; I = 1B , I = I - 1 , I = 1A
jmp    k5

Y44:   cmp    bl, 'Y'
jne    Z44

```

```

mov ax, 1Bh
mov [I], ax          ; I = 1C , I = I - 1 , I = 1B
jmp k5

```

```

Z44:    cmp bl, 'Z'
        mov ax, 1Ch
        mov [I], ax          ; I = 1D , I = I - 1 , I = 1C

```

k5: ;----- Anahtara Karşılık Gelen Adresin Hesaplanması -----  
adres1:;----- MASTER.DAT Dosyası için Adres Hesabı -----

```

;*****
;*      RecNo = (G-1)*349h +(H-1)*1Dh +(I-1) +1      *
;*      RecNo = (G-1)*841  +(H-1)*29  +(I-1) +1      *
;*****
;

```

```

mov ax, 00349h ;
mul [G]        ; 1.karakterin sayısal değeri
mov [AA11X], ax ; Çarpımın düşük baytının saklandığı yer

```

```

mov ax, 0001Dh ; Kayıt edilecek bilgi 100 bayt + Null
mul [H]        ; 2.karakterin sayısal değeri
mov [AA22X], ax ; Çarpımın düşük baytının saklandığı yer

```

```

mov ax, [AA11X] ; Çarpımların düşük baytlarının
add ax, [AA22X] ; toplanması
add ax, [I]     ;
add ax, 1h      ; RecNo = ax

```

```

mul [RecSize] ; Address = RecNo * RecSize

```

```

mov [MADRESL], ax ; Çarpımın düşük baytının sak. yer
mov [MADRESH], dx ; Çarpımın yüksek baytının sak. yer

```

adres:

;----- KEY alanına MASTER.DAT dosyasının adreslerinin yerleştirilmesi

```

mov di, OFFSET key ; Key'in uzunluğunu
call StrLength      ; bul

```

```

mov di, OFFSET key ; Key'in bitiş
mov ax, di          ; konumunu
add ax, cx          ; tesbit
mov di, OFFSET key ; et
mov di, ax          ;

```

```

mov ax, [MADRESL] ; MASTER dosyanın Yüksek Adresini
stosw             ; yükle

```

```

        mov ax, [MADRESH]      ; MASTER dosyasının Yüksek Adresi

;----- Adresi düzelt

        cmp ah, 002h           ; ah>=2 ise ah=ah+30 yap
        jbe ilave1
        jmp ilave
ilave1:  add ah, 030h           ; ah=ah+30h

ilave:   cmp al, 002h           ; al>=2 ise al=al+30 yap
        jbe ilave2
        jmp yerles
ilave2:  add al, 030h           ; al=al+30h

yerles:  stosw                  ; yükle

        call NewLine           ; Anahtar ve arkasına
        mov di, OFFSET key      ; yerleştirilen adres
        call StrWrite           ; bilgisini ekrana yaz

;----- BigBuffer'a ANAHTAR+ADRES bilgisini yerleştir -----

        mov si, OFFSET key
        mov di, OFFSET BigBuffer
        pop dx
        call StrInsert
        push dx

mas;;----- MASTER.DAT için File Pointer'ın Ayarlanması -----

        mov ax, [MADRESL]      ; Düşük adresin
        mov dx, ax             ; dx'e yüklenmesi
        mov ax, [MADRESH]      ; Yüksek adresin
        mov cx, ax             ; cx'e yüklenmesi

        mov ah, 042h           ; DOS seek-file (dosya-araştırma fonk.)
        mov al, 0              ; Dosyanın başlangıcından araştır
        mov bx, [handle1]      ; Assign file handle to bx
        int 21h

        mov ax, 00000h
        mov [MADRESL], ax
        mov [MADRESH], ax

;----- MASTER.DAT Dosyasına Bilgi Yazma -----

        mov bx, [handle1]      ; OFFSET FileName
        mov cx, 65h            ; Kayıt edilecek bil.uzunluğu(100byte)

```

```

mov  ah, 40h
mov  dx, OFFSET testString
int  21h

```

;————— Başka kayıt var mı.....?

```

mov  di, OFFSET mesaj1      ;
mov  al, [harf1]             ; İlk iki karakteri
stosb                        ; ekrana
cld                          ; yaz
mov  al, [harf2]             ;
stosb                        ;

call  NewLine                ; İlk iki karakteri
mov  di, OFFSET mesaj1      ; belirtilen başka kayıt
call  StrWrite               ; varmı mesajını yaz

mov  ah, 01                  ; Klavyeden bir karakter
int  21h                     ; oku ekrana yaz

cmp  al, 'E'
je   wwwww
cmp  al, 'e'
je   wwwww
cmp  al, 'H'
je   cik1
cmp  al, 'h'
je   cik1
jmp  cik1

```

```

wwwwww:  mov  di, OFFSET key      ; Key'in uzunluğunu
          call StrLength          ; bul
          pop  dx                 ; BigBuffer'in indeksini geri çağır
          mov  ax, cx             ; Key'in uzunluğu ile indeksi topla
          add  ax, dx             ; yeni indeksi elde et.
          mov  dx, ax             ;
          push dx                 ; Yeni indeksi sakla
          jmp  wwwwww

```

cik1:

;————— Anahtarın 1. ve 2. karakterlerinin bl ve cl'ye yüklenmesi

```

mov  bl, [harf1]
mov  cl, [harf2]

```

## ;----- 1.Karakterin bulunması -----

```

push ds          ; Programın başlanğıç
push es          ; adresinin saklanması

mov ax, 0D000h   ; Ön belleğin segment adresi
mov ds, ax
mov si, 00000h   ; Ön belleğin offset adresi

aqq: lodsb        ; Ön bellekten 1 baytlık karakter bilgi. okunması
      cmp al, bl   ; Anah.1.karak. ile okunan karakteri karşılaştır
      je found11
      mov ax, si    ; Ön bellekten okunacak bir sonraki karakter
      add ax, 00002h ; için adresin belirlenmesi
      mov si, ax    ; Yeni offset adresinin yüklenmesi
      jmp aqq

found11: lodsw      ;2. karakterlerin başlanğıç adresinin bulunması
          mov si, ax ; Offset adresinin yüklenmesi

```

## ;----- 2.Karakterin bulunması -----

```

bqq: lodsb        ; Anahtarın 2.karakteri ile ön bellekten okunan
      cmp al, cl   ; karakterlerin karşılaştırılması
      je found22
      mov ax, si    ; Ön bellekten okunacak bir sonraki karakter
      add ax, 00006h ; için, offset adresinin belirlenmesi
      mov si, ax    ; Yeni offset adresinin yüklenmesi
      jmp bqq

found22: lodsw
          mov cx, ax ; KEY.DAT'in yüksek adresinin cx'e yüklenmesi
          cld
          lodsw
          mov dx, ax ; KEY.DAT'in alçak adresinin dx'e yüklenmesi
          cld
          lodsw
          mov bx, ax ; Aramanın yapılacağı bellek alanının uzunluğu

```

```

pop es      ;
pop ds      ;

```

```

mov [ADRESH], cx ; KEY.DAT'in yüksek adresinin saklanması
mov [ADRESL], dx ; KEY.DAT'in alçak adresinin saklanması
mov [MAXAREA], bx ; Aramanın yapılacağı bel.al.uz.saklanması

```

;----- KEY.DAT için File Pointer'ın Ayarlanması -----

```

mov ax, [ADRESL]      ; Düşük adresin
mov dx, ax             ; dx'e yüklenmesi
mov ax, [ADRESH]      ; Yüksek adresin
mov cx, ax             ; cx'e yüklenmesi

mov ah, 042h          ; DOS seek-file (dosya-arastırma fonk.)
mov al, 0              ; Dosyanın başlangıcından araştır
mov bx, [handle]      ; Assign file handle to bx
int 21h

```

;----- KEY.DAT Dosyasına Bilgi yazma (KEY+ADRES) -----

```

mov bx, [handle]      ; OFFSET FileName
mov cx, 0008Dh        ; 141 Byte, 14 bayt bir KAYIT uzunluğu
mov ah, 40h
mov dx, OFFSET BigBuffer
int 21h

mov di, OFFSET BigBuffer
call StrNull

```

;----- KEY.DAT ve MASTER.DAT Dosyasını Kapama -----

```

call NewLine          ; " Başka kayıt
mov di, OFFSET kayıt  ; girişi varmı" mesajını
call StrWrite          ; ekrana yaz

mov ah, 01            ; Klavyeden bir karakter
int 21h               ; oku ekrana yaz

cmp al, 'E'
je wwwww0
cmp al, 'e'
je wwwww0
cmp al, 'H'
je cik17
cmp al, 'h'
je cik17
jmp cik17

```

wwwww0: jmp wwwwwwr

```

cik17:  mov ah, 3Eh          ; KEY.DAT dosyasını
        mov bx, [handle]    ; kapat
        int 21h            ;

```

```

mov  ah, 3Eh          ; MASTER.DAT dosyasını
mov  bx, [handle1]    ; kapat
int  21h              ;

call  NewLine         ;
mov  di, OFFSET don   ;
call  StrWrite        ; Ana menüye dönüş
mov  ah, 01h          ;
int  21h              ;
jmp  Start1

```

oku:;===== Dosyadan Bilgi Okuma işleminin Başlangıcı =====

;----- Okuma için anahtar kelimenin alınması -----

```

mov  ah, 3Dh          ; KEY.DAT dosyasını
mov  al, 0            ; yalnızca okuma
mov  dx, OFFSET KeyFilename ; için aç
int  21h              ;
mov  [handle], ax     ;

mov  ah, 3Dh          ; MASTER.DAT dosyasını
mov  al, 0            ; yalnızca okuma
mov  dx, OFFSET FileName ; için aç
int  21h              ;
mov  [handle1], ax    ;

```

```

oku0:  mov  di, OFFSET key1
       call StrNull

       call NewLine
       mov  di, OFFSET mkey1 ; 'Okuma için anahtar kelimeyi
       call StrWrite         ; yazın' mesajı

       mov  di, OFFSET key1 ; Anahtar kelimenin
       mov  cx, keymax       ; 10 bayt
       call StrRead          ; klavyeden alınması

```

;----- Anahtarın 1. ve 2. karakterlerinin bl ve cl'ye yüklenmesi

```

mov  si, OFFSET key1 ; Anahtarın 1.karakterini
lods [key1]           ; BL'ye yükle
mov  bl, al           ;
cld                   ; si←si+1, di←di+1
lods [key1]           ; Anahtarın 2.karakterini
mov  cl, al           ; CL'ye yükle

```

----- 1.Karakterin bulunması -----

```

push ds          ; Programın başlangıç
push es          ; adresinin saklanması

mov ax, 0D000h   ; Ön belleğin segment adresi
mov ds, ax
mov si, 00000h   ; Ön belleğin offset adresi

aq:  lodsb        ; Ön bellekten 1 baytlık karakter bilgi. okunması
     cmp al, bl   ; Anah.1.karak. ile okunan karakteri karşılaştır
     je found1
     mov ax, si    ;Ön bellekten okunacak bir sonraki karakter
     add ax, 00002h ; için adresin belirlenmesi
     mov si, ax    ; Yeni offset adresinin yüklenmesi
     jmp aq

found1: lodsw      ;2. karakterlerin başlangıç adresinin bulunması
        mov si, ax ; Offset adresinin yüklenmesi

```

----- 2.Karakterin bulunması -----

```

bq:  lodsb        ; Anahtarın 2.karakteri ile ön bellekten okunan
     cmp al, cl   ; karakterlerin karşılaştırılması
     je found2
     mov ax, si   ; Ön bellekten okunacak bir sonraki karakter
     add ax, 00006h ; için, offset adresinin belirlenmesi
     mov si, ax    ; Yeni offset adresinin yüklenmesi
     jmp bq

found2: lodsw
        mov cx, ax ; KEY.DAT'ın yüksek adresinin cx'e yüklenmesi
        cld
        lodsw
        mov dx, ax ; KEY.DAT'ın alçak adresinin dx'e yüklenmesi
        cld
        lodsw
        mov bx, ax ; Aramanın yapılacağı bellek alanının uzunluğu

        pop es      ;
        pop ds      ;

        mov [ADRESH], cx ; KEY.DAT'ın yüksek adresinin saklanması
        mov [ADRESL], dx ; KEY.DAT'ın alçak adresinin saklanması
        mov [MAXAREA], bx ; Aramanın yapılacağı bel.al.uz.saklanması

```

;----- KEY.DAT dosyasının Okunması İçin -----

adres111:;--- File Pointer'in Ayarlanması -----

```

        mov     ax, [ADRESL]      ; Düşük adresin
        mov     dx, ax            ; dx'e yüklenmesi
        mov     ax, [ADRESH]      ; Yüksek adresin
        mov     cx, ax            ; cx'e yüklenmesi

        mov     ah, 042h          ; DOS seek-file (dosya-araştırma fonk.)
        mov     al, 0             ; Dosyanın başlangıcından araştır
        mov     bx, [handle]       ; Assign file handle to bx
        int     21h
        jmp     tyoku
        jc      handerror1
        jmp     tyoku
handerror1: mov     di, OFFSET dgoster1
              call    StrWrite

```

;----- Yeni Pozisyonda KEY.DAT Dosyasından 141 bayt'ın okunması--

```

tyoku:   mov     di, OFFSET BigBuffer
        call    StrNull

        mov     ah, 3Fh           ; DOS Dosya-okuma fonksiyonu
        mov     bx, [handle]       ; BX'e dosya mandalının tanıtılması
        mov     cx, 0008Dh        ; 0008Dh Okunacak bilgi 141 byte
        mov     dx, OFFSET BigBuffer ; Gidilecek adres
        int     21h

        call    NewLine           ; Okunan bloğu ekrana yaz
        mov     di, OFFSET BigBuffer
        call    StrWrite

```

;----- Okunan Blok İçinde Anahtarın Tamamının Bulunması -----

```

        mov     si, OFFSET key1    ; Aranılacak alt dizinin baş.adresi
        mov     di, OFFSET BigBuffer ; İçinde aramanın yap.dizinin baş ad.
        call    StrPos             ; Aramanın yapılması

        jz      foundkey
        jmp     notfound

foundkey: mov     di, OFFSET BigBuffer
        mov     ax, di              ; Key'in indeksini ax'e yükle
        add     ax, dx              ;
        push    ax
        mov     di, OFFSET key1

```



```

        mov     bx, [handle1]           ; Assign file handle to bx
        int     21h
        jc      handerror2
        jmp     oku88

handerror2: mov     di, OFFSET dgoster2
            call    StrWrite
            jmp     wwwww9

oku88:   mov     ax, 00000h
            mov     [MADRESL], ax
            mov     [MADRESH], ax

;----- Dosyadan bilgi okuma -----

        mov     di, OFFSET testString
        call    StrNull

        mov     ah, 3Fh
        mov     bx, [handle1]
        mov     cx, 65h                ; Okunacak bilgi 100 byte
        mov     dx, OFFSET testString
        int     21h

        call    NewLine                ; Kayıtın ekrana
        mov     di, OFFSET testString ; yazdırılması
        call    StrWrite                ;

;----- Başka kayıt varmı ? -----

        call    NewLine                ; Başka okunacak kayıt varmı?
        mov     di, OFFSET okuma       ; mesajını ekrana yazma
        call    StrWrite                ;

        mov     ah, 01                ; Klavyeden bir karakter oku
        int     21h                    ; ekrana yaz

        cmp     al, 'E'
        je      qqqqq9
        cmp     al, 'e'
        je      qqqqq9

        cmp     al, 'H'
        je      wwwww9
        cmp     al, 'h'
        je      wwwww9

```

```
qqqqq9:    jmp    oku0
```

```
;----- Dosyaları kapatıp ana menüye dönüş -----
```

```
www9:      mov    bx, [handle]          ; KEY.DAT dosyasını
            mov    ah, 3Eh              ; kapat
            int     21h                 ;

            mov    bx, [handle1]        ; MASTER.DAT dosyasını
            mov    ah, 3Eh              ; kapat
            int     21h                 ;

            call   NewLine              ;
            mov    di, OFFSET don       ;
            call   StrWrite              ; Ana menüye dönüş
            mov    ah, 01h              ;
            int     21h                 ;
            jmp    Start1               ;
```

```
Exit:      mov    ah, 04Ch              ; Programı bitir ve
            mov    al, [exitCode]       ; DOS ortamına
            int     21h                 ; dön
            END     Start                ; Programın sonu
```