C64 TÜRKİYE

SAYI : #$08 BİLGİ PAYLAŞTIKÇA ARTAR EKİM 2005

BASIC Öğreniyorum #1
HEX ve ASCII Memory Dump
1x1 Text Fader
1x2 Karakter Seti
Dünden Bugüne Ülkemizde Çıkan C64 Dergileri
Program Dökümleri

hades@amigaturk.com

BASIC ÖĞRENİYORUM (DERS 1)

Hazırlayan: Burak ÇETİNDAĞ İletişim : criminal@amigaturk.com

Herkese merhaba bu sayımızdan itibaren eski dostumuz C64 ile konuşabileceğimiz bir dil olan BASIC derslerine başlayacağız (1 dil bir insan 2 dil insan) bu gıcık espriyi yaptıktan sonra dersimize geçelim öncelikle bilinmesi gereken bilgisayarın temel yapısı bu yapıyı çözdükten sonra BASIC öğrenmek çok kolaylaşacaktır zira programcı olmanın temeli budur. Bilgisayarın 5 ana ünitesini inceleyelim

1) Giriş ünitesi (INPUT)
2) Çıkış ünitesi (OUTPUT)
3) Bellek Ünitesi (MEMORY)
4) Aritmetik Mantık Ünitesi (ARITHMETIC LOGIC UNIT A.L.U)
5) Yönetim Ünitesi (CENTRAL PROCESSING UNIT C.P.U)

Bilgisayar elektronik bir makinedir diyoruz. O halde nasıl oluyorda bir dizi aritmetik işlem alfabetik bilgi akımı mümkün oluyor? Temelde iki çeşit elektrik sinyalinden bahsetmekteyiz.Bunlar düşük voltajlı sinyal ve yüksek voltajlı (genelde 5 volt) sinyalleridir.Elektrik dilinde iki nokta arasında potansiyel enerji farkı varsa voltaj farkı da vardır. Bu prensibe dayanarak "bit" ler dediğimiz 0 ve 1'leri oluşturmak mümkündür. Eğer voltaj farkı sıfır volt ise 0 "bit" ini voltaj farkı 5 volt ise 1 "bit" ini oluşturmuş oluruz. "Bit" lerin 8 tanesi bir arada olursa 1 "byte" denir. Bütün harfleri bir dizi 0 ve 1 leri bir araya getirerek yani kodlama sistemini kullanarak oluşturmak mümkündür.

PROGRAMLAMAYA GİRİŞ

Bilgisayarı programlanabilen bir makine olarak düşünmek gerekir. Yapılan işler hızlı ve hacim itibariyle kabarıkta olsa, programı basamak,basamak alırsanız her işlemin çok basit olduğunu görürsünüz. İlk programı küçük bir hesap makinesi ile de rahatça yapılabilecek bir işlem üzerinde kuralım. Bilgisayara iki tane rakam verelim ve bu rakamları toplayarak neticesini ekrana yazmasını isteyelim.

```
10 INPUT A
20 INPUT B
30 LET TP=A+B
40 PRINT TP
50 END
```

Yukarıdaki programa bakınca ilk dikkat edilen özellik, programın satırlardan oluşması ve her satırın da bir numara ile başlamasıdır: örneğin 10,20.... gibi. Bu numaralara satır numaraları (Statement Number) denir. Satır numaraları küçükten büyüğe doğru artar. Bu düzenlemeyi bilgisayarın kendisi yapmaktadır. Satır numaraları 1 ile 63999 arasında tam sayı ve pozitif olan sayılardır. Programın akışına göre düzenlenmelidirler. Yani A ve B nin değerini vermeden, toplamı oluşturan TP değerini hesaplamak mümkün olamayacağından, programdaki 30 no'lu satıra 10 numarası vermekle çalışabilen bir program yapmak mümkün değildir. Dikkat edilecek bir diğer özellikte satır numaralarının onar,onar artmış olmasıdır. Bu numaralar birer,birer arttırılabilirlerdi ve programda pekala çalışırdı. Bu değerin onar,onar arttırılmasındaki amaç eğer ileride araya başka satırlar ilave etmek gerekirse programın genel akışını bozmadan bunu gerçekleştirmek içindir. Aksi halde ilave edilen satırdan sonraki satırların numaralarını tekrar düzenlemek gerekecektir. Demek ki programın genelinde bir değişiklik yapmadan 10 no'lu satırla 20 no'lu satır arasına 11,12,13,14,15,16,17,18,19 nolu satırları ilave etmek mümkündür.

INPUT , LET , PRINT komutları

10 numaralı satırı biraz daha detaylı inceleyelim.

INPUT A : input komutu ile A basit değişkeninden oluşmaktadır. **INPUT** komutu (girdi) anlamına gelmektedir. Bilgisayar dilinde **INPUT** kendisinden sonra gelen değişkenin, bu durumda A nın sayısal değerini klavye yolu ile vermek için kullanılır. Programın tamamını bilgisayarın hafızasına girdikten sonra programı çalıştırmak için RUN komutunu yazıp RETURN tuşuna basmak gerekir.Bu durumda bilgisayar ilk olarak **10 INPUT A** satırını

okuyacaktır. Satırda **INPUT** komutu olduğundan bizden A değişkenini isteyecektir. Bunu belirtmek için de ekrana bir soru işareti yazar. A değişkeninin değeri 5 olsun şimdi 5 yazıp **RETURN** tuşuna basın. Bu işlemi yaparken yazılan 5 rakamını ekranda görmek mümkündür. İleri de göreceğimiz gibi bilgisayar A değişkeni için hafızasında ayırdığı yere 5 rakamını yerleş-tirecektir.

Bilgisayar daha sonra ikinci satır olan **20 INPUT B** satırına geçecektir. Burada ki durum da ilk satıra benzediği için ekranda soru işareti görülecektir. Bu demektir ki bilgisayar B değişkeninin değerini istemektedir. B değişkeninin değeri 3 olsun 3 yazıp **RETURN** tuşuna basarak bu işlemi de tamamlayalım. Bilgisayar bir sonraki satırda bilgileri değerlendirmeye başlayacaktır. **30 LET TP = A + B** satırında **LET** komutu kendinden sonraki verilen ifadenin geçerli olmasını mümkün kılan bir deyimdir.Burada TP denilen bir değişken daha vardır. Hemen belirteyim, değişkenlerin bir harf ile edilmesi şartı yoktur. A,B değişkenler olduğu gibi TP de bir değişkendir.

TP = A + B ifadesi ile bilgisayara deniliyor ki A değişkeninin değeri ne ise onu al B değerinin değişkeni ne ise onu da al, topla ve bu toplamı TP değişkeni ile ifade edilen hafızaya yerleştir. Bu da **LET** komutunu kullanılarak yapılır. Bazı bilgisayarlarda **LET** komutunu kullanmaya gerek yoktur. O zaman bu satır **30 TP =A + B** şeklindedir. O halde 30 numaralı satırdan sonra bilgisayar TP adında bir değişken oluşturdu. Bu değişkene ait olan hafızada A ve B nin sayısal toplamı olan 8 vardır.

40 PRINT TP satırında ki **PRINT** komutu kendisinden sonra gelen değişkenin sayısal değerini ekrana yazmak için kullanılır. Bilgisayar ekrana bağlı değil de printer a bağlı ise TP değişkeninin sayılsa değeri bu defa kağıt üzerine yazılacaktır. Ekran kullandığımıza göre demek ki ekrana 8 rakamı yansıyacaktır.

50 END satırındaki **END** komutu programın sonu geldiğini gösterir. Her programın sonunda **END** komutunu kullanmak gerekir.

PROGRAMIN ÇALIŞTIRILMASI

Her satırı yazıp hafızaya yerleştirdikten sonra programı çalıştırmak için **RUN** komutunu yazmak gerekir. Şimdi **RUN** yazın ekranda çıkacak işlem aşağıdaki gibi olacaktır.

```
? 5
? 3
8
```

5 ile 3 ün toplamı 8 dir. Bilgisayar daha sonra hazır anlamına gelen **READY** mesajını yazarak kontrolü kullanıcıya bırakır.

LIST, Satır Numaraları ve Program Akışı

Daha öncede belirttiğim gibi satır numaraları küçükten büyüğe doğru artar. Bu artış onar, onar olabileceği gibi birer,birer de olabilir. Satır numaraları numaraları programın akı-şına göre düzenlenmelidirler. Numaralama yanlış yapıldığında aşağıdaki örnekte olduğu gibi yanlış netice alınacaktır.

Bilgisayar bu satırları hafızasına aldıktan sonra satır numaralarına göre sıraya koymaktadır.Hafızada ne olduğunu görmek için yani programı listelemek için **LIST** deyimi kullanılır.Şimdi **LIST** yazıp **RETURN** tuşuna basalım

```
10 INPUT A
15 LET TP=A+B
30 INPUT B
40 PRINT TP
50 END
```

Bu programa **RUN** demek mümkün olmayacaktır. Sebebi ise bilgisayar 15 no'lu satıra geldiği zaman B değişkeninin değerini bilmediği için toplama işlemini yapamayacaktır. (HADES'in notu : C64'te eğer sayısal bir değişkene herhangi bir değer verilmeden o değişkenle işlem yapılması durumunda değişkenin değeri sıfır kabul edilerek işlem yapılır.) O halde programdaki satırlara numara verirken, numaralar problemin çözümü için gerek işlem sayısına paralel olarak artmalıdır.Bence en iyisi programı yazarken önce satır numaralarını küçükten büyüğe doğru sıralamalı sonra yazı kısımlarını işlem sırasına göre doldurmalıdır.

Evet arkadaşlar bu aylık benden bu kadar önümüzdeki dersimizde **BASİT DEĞİŞKENLER (Aritmetik Operatörler,İfadeler ve Komparatörler) REMARK** ve **PRINT** komutlarını inceleyip bol, bol örnekler göreceğiz. Buraya kadar gördüğünüz gibi BASIC dili çok kolay anlaşılır bir dildir her ne kadar günümüzde pek kullanılmasada bazen bize gerekebilecek ve en önemlisi 68K işlemcili bilgisayarlarda süper iş görecek sade ve stabil bir dildir. Önümüzdeki sayıda görüşmek üzere **BİR COMMODORE ALIN ONUNLA KALIN** ☺

HEXADECIMAL VE ASCII MEMORY DUMP

```
;=============================================================================================;
; hexadecimal ve ascii memory dump programı                                                   ;
; C64 TÜRKİYE dergisi sayı 8 için hazırlanmıştır                                              ;
; 10-05-2005 / ismail "hades" şahin                                                           ;
; hades@amigaturk.com  www.c64turkiye.com  www.amigaturk.com  c64turkiye@amigaturk.com        ;
;=============================================================================================;

                *=$0801

prg             .word nextline
                .word 2005
                .byte $9e
                .text "2061"
                .byte 0
nextline        .word 0
```

start etiketli yere kadar olan kısımda LIST komutuyla göreceğiniz 2005 SYS2061 satırını oluşturuyoruz.

```
start           lda   #$0d              ;yeşil renk kodu
                jsr   $e536
                dex
                stx   $d020
                stx   $d021
                stx   $fb
                stx   $fc
```

Yukarıdaki bölümde karakter rengi yeşil olacak şekilde ekranı siliyoruz ve ekran silindikten sonra X registerinin değerini 1 azaltarak "0" yapıyoruz. Bu değeri $D020 ve $D021 adreslerine vererek ekran renklerini siyah yapıyoruz. Ayrıca daha sonra kullanacağımız $FB ve $FC adreslerini sıfırlıyoruz.

```
                lda   #<text1
                ldy   #>text1
                jsr   $ab1e
                lda   #$3a
                jsr   $ffd2
                lda   #$24
                jsr   $ffd2
```

Önce ekrana biraz yazı yazıyoruz sonra ":" ve "$" karakterlerini basıyoruz.

```
adres_gir       jsr   hex_byte_in
                sta   $fc               ;high bayt
                jsr   hex_byte_in
                sta   $fb               ;low bayt
```

Bu bölümde ilk önce adresin HIGH baytını klavyeden girip $FC adresine saklıyoruz. Sonrada LOW baytını girip $FB adresinde saklıyoruz.

```
                ldy   #$00
                ldx   #$06
                clc
                jsr   $fff0
```

Yukarıdaki komutlarla kursörün yerini belirliyoruz. Y registerine kursörü koyacağımız yerin sütun değerini, X registerine ise satır değerini vererek Kernal Romdaki PLOT rutinine sıçrıyoruz.

```
                lda   #<text2
                ldy   #>text2
                jsr   $ab1e
                lda   #<text3
                ldy   #>text3
                jsr   $ab1e
```

Ekrana birşeyler yazdırıyoruz ve asıl bölüme geçiyoruz.

```
dumper          lda   #$10
                sta   line
```

Ekranda bir seferde kaç satır gösterileceğini ayarlıyoruz.

```
line_dump       lda   #$20
                jsr   $ffd2

                lda   $fc
                jsr   mem_to_scr
                lda   $fb
                jsr   mem_to_scr
                lda   #$3a
                jsr   $ffd2
                lda   #$20
                jsr   $ffd2
```

Önce bir adet "boşluk" basıyoruz. Programın başında girdiğimiz adresin önce HIGH baytını, sonrada LOW baytını hex olarak ekrana yazdırıyoruz. Son olarak ":" ve "boşluk" karakterlerini basıyoruz

```
one_line_dump   ldy   #$00
loop1           lda   ($fb),y
                sta   temp_scr,y
                jsr   mem_to_scr
                lda   #$20
                jsr   $ffd2
                iny
                cpy   #$08
                bne   loop1
```

Bu bölümde bir satıra hex dump yapıyoruz. Y registerini sıfırlayıp bayt sayacı olarak kullanıyoruz. LDA komutuyla hafızadan o anki baytı alıyoruz ve geçici olarak saklıyoruz. Hafızadan okuduğumuz baytı JSR komututla HEX formatlı olarak ekrana yazdırıyoruz. Ekrana bir baytı yazdırdıktan sonra bir adet "boşluk" basıyoruz. Y registerini (bayt sayacı) 1 arttırıp 8 ile karşılaştırıyoruz. 8 olmadıysa hafızadaki bir sonraki bayt için LOOP1 etiketli yere sıçrıyoruz. 8 olduysa program aşağıdan devam ediyor.

```
ascii           ldy   #$00
ascii_dump      lda   temp_scr,y
                cmp   #$20
                bcc   no_ascii
                cmp   #$60
                bcs   no_ascii
                .byte $2c
no_ascii        lda   #$2e
                jsr   $ffd2
                iny
                cpy   #$08
                bne   ascii_dump
```

Burada ise daha önce hafızadan okuyup geçici olarak sakladığımız baytların varsa ascii karşılığını ekrana yazdırıyoruz. İlk olarak Y registerini sayaç olarak kullanmak üzere sıfırlıyoruz. Sakladığımız ilk baytı alıyoruz. Önce "boşluk" karakterinden küçük olup olmadığını kontrol ediyoruz. Eğer küçükse "no_ascii" yazan yere gidiyoruz ve ekrana

"." karakterini basıyoruz. Eğer okunan bayt "boşluk" karakterinden büyükse bu sefer (←) karakterinden büyük olup olmadığını kontrol ediyoruz. Eğer büyükse yine "no_ascii" yazan yere gidiyoruz. Eğer küçükse yani okunan bayt $20-$60 arasındaysa ekrana basıyoruz. Fakat burada $2C bayt'ı bilgisayarı kandırmak için kullanıyoruz. $2C değeri "BIT" komutunun kodu olup kullandığı parametre 2 baytlıktır. Bu parametreler ise "no_ascii" yazan yerdeki "LDA #$2E" komutunun baytlarıdır. Dolayısıyla "BIT $A92E" gibi bir komut yazmış oluyoruz. Bu durumda komut sırası .... BCS NO_ASCII - BIT $A92E - JSR $FFD2 şeklinde olmaktadır. Dolayısıyla Aküdeki değer değişmemiş olmakta ve bu değerin ASCII karşılığını yazdırmış oluyoruz. Normalde "no_ascii" yazan yere gelindiğinde Aküye $2E değeri yüklenmektedir ve $2E'nin ascii karşılığı "." karakteridir. Bu nedenle hafızadan okunan baytların ascii karakter karşılığı yoksa "." basılmaktadır. Bu işlemleri 8 defa tekrarlıyoruz ve sonra program aşağıdan devam ediyor.

```
new_line        lda   #$0d
                jsr   $ffd2
```

Yukarıdaki iki komutla bir alt satıra geçiyoruz.

```
mem_setup       lda   $fb
                clc
                adc   #$08
                sta   $fb
                bcc   no_inc_fc
                inc   $fc
no_inc_fc       dec   line
                bne   line_dump
```

Bu bölümde ilk olarak hafızadan okuyacağımız yeni yerin adresini hesaplıyoruz. $FB adresinde okuma yaptığımız adresin LOW baytı bulunmaktadır. Bu adresteki değeri 8 ile toplayıp geri yazıyoruz. (Her seferinde 8 bayt okuduğumuz için 8 ile topluyoruz). Toplama sonucunda elde biti varsa (yani sonuç 255'ten büyükse -"C" biti "1" ise-) bu sefer $FC adresindeki HIGH baytı bir arttırıyoruz. Eğer elde biti "0" ise yani sonuç 255 veya küçükse "no_inc_fc" yazan yere gidiyoruz. Burada ise daha önce tanımlamış olduğumuz ekranda bir seferde gösterilecek satır sayısını bir azaltıyoruz. "0" olmadıysa "line_dump" yazan yere gidip ekranda 16 satır gösterilinceye kadar işlemleri tekrarlıyoruz.

```
f_keys          jsr   $ffe4
                cmp   #$85        ;F1 - Sonraki sayfa
                beq   next_page
                cmp   #$86        ;F3 - Önceki sayfa
                beq   old_pages
                cmp   #$87        ;F5 - Yeni adres
                beq   restart
                cmp   #$88        ;F7 - Çıkış
                bne   f_keys
                jmp   $a474       ;Basic'e geri dön
restart         jmp   start
next_page       .byte $2c
old_pages       dec   $fc
                ldy   #$00
                ldx   #$08
                clc
                jsr   $fff0
                jmp   dumper
```

Burası ekranda 16 satır yani 16*8=128 bayt dump edildikten sonra basılması gereken "F(onksiyon)" tuşlarının kontrol edildiği bölümdür. Önce bir JSR komutuyla tuşları okuyoruz ve "F" tuşları olup olmadığını kontrol ediyoruz. "F1" ile "next_page" yazan yere gidiyoruz. Daha önce bahsettiğimiz şekilde "BIT" komutuyla bilgisayarı kandırıyoruz ve kursörün yerini belirleyip dump etme işine tekrar başlıyoruz. "F3" ile daha önceki sayfayı ekrana getiriyoruz. Bunun için o anki hafıza adresinin HIGH baytının tutulduğu $FC adresinin değerini 1 azalttıktan sonra kursörün yerini belirliyoruz ve başa dönüyoruz. "F5" ile farklı bir adresi dump etmek için programın en başına zıplıyoruz. "F7" ile programdan çıkıyoruz.

Burası altrutinlerin olduğu bölümdür.

```
hex_byte_in     jsr   hex_key_in        ;high nibble
                jsr   key_to_mem
                asl
                asl
                asl
                asl
                sta   $fb
                jsr   hex_key_in        ;low nibble
                jsr   key_to_mem
                ora   $fb
                sta   $fb
                rts
```

Klavyeden bir baytlık hex sayı girme ve bu sayıyı hafızaya saklama bölümüdür. Önce klavyeden hex karakter girilen alt rutine gidiyoruz. Geri dönüşte aküde bulunan tuş değerinin sayısal karşılığını almak için bir başka alt rutine gidiyoruz. Buradan geri dönüşte ise elimizdeki sayıyı 4 kez sola kaydırıp high nibble olarak $FB adresine saklıyoruz.Aynı alt rutinlere bir kez daha uğruyoruz ve daha önce high nibble olarak sakladığımız değerle OR işlemi yaptıktan sonra aynı adrese yazıyoruz.

```
hex_key_in      jsr   $ffe4
                ldx   #$00
control         cmp   key_table,x
                bne   next
                jsr   $ffd2
                rts
next            inx
                cpx   #$10
                bne   control
                jmp   hex_key_in
```

Bu kısımda klavyeden sadece hex sistem tuşlarına basılıp basılmadığını kontrol ediyoruz. Önce klavye taranıyor. Tarama sonunda Aküdeki değeri bir sayaca göre tablodaki değerle karşılaştırıyoruz. Eğer karşılaştırma sonucunda basılan tuş tabloda varsa karakter ekrana basılıyor ve rutinden çıkılıyor. Yoksa klavye tekrar taranıyor.

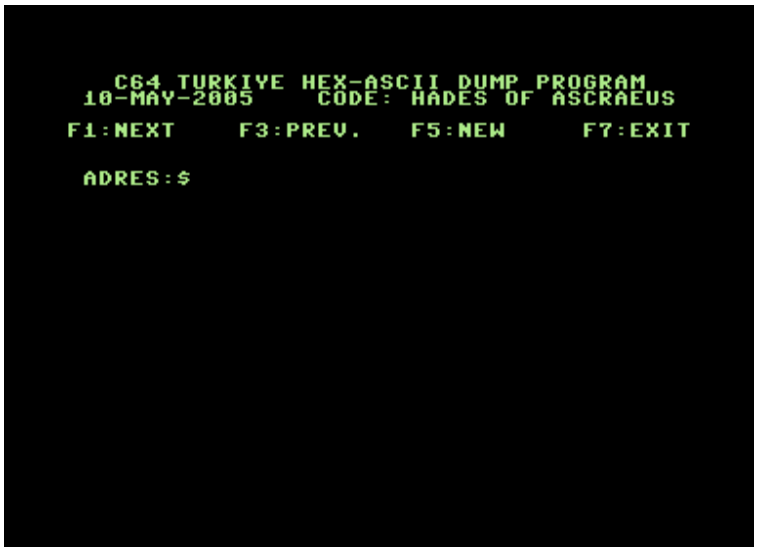
```
key_to_mem      ldx   #$00
loop0           cmp   key_table,x
                beq   val_read
                inx
                cpx   #$10
                bne   loop0
val_read        lda   val_table,x
                rts
```

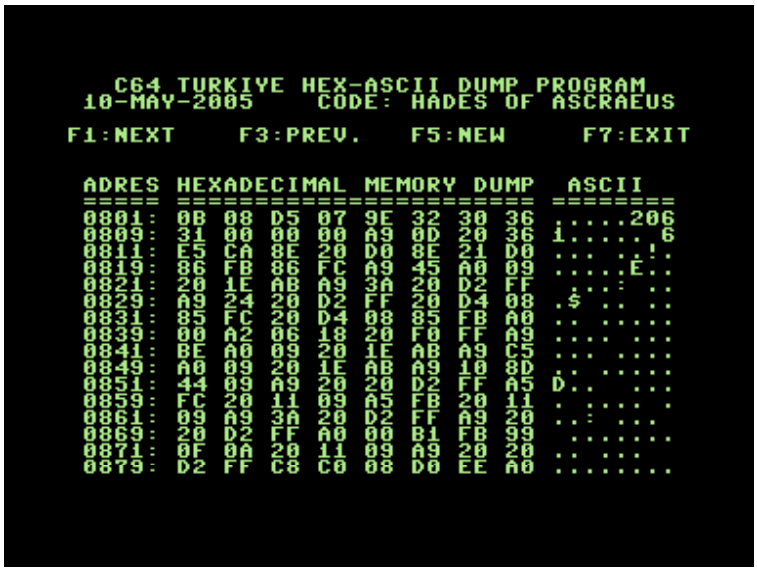
Bu kısımda basılan tuşa göre hafızaya alınacak sayı elde ediliyor. Aküdeki değer bir tablo ile karşılaştırılıyor. Karşılaştırma sonucu eşitlik varsa, kullanılan sayacın değeri kullanılarak başka bir tablodan hafızaya yazılacak değer okunuyor ve rutinden çıkılıyor.

```
mem_to_scr      pha
                lsr
                lsr
                lsr
                lsr
                jsr   convert
                pla
                and   #$0f
convert         tax
                lda   key_table,x
                jsr   $ffd2
                rts
```

Geldik son alt rutine. Burada yapılan iş hafızadan okunan bir baytı hex olarak ekrana yazdırmak. İlk olarak hafızadan okunan bayt daha sonra tekrar kullanılmak üzere yığına atılıyor. Sonra HIGH nibble değerini ekrana yazdırmak için 4 kez sağa kaydırılıyor. Elde edilen değer sayaç olarak kullanılarak, sayaca göre tablodan okunan değer ekrana yazdırılıyor.Daha sonra yığındaki değer geri alınarak $0F sayısı ile AND işlemi yapılıyor. Böylece LOW nibble elde ediliyor. Sayaç işlemleri bir kez daha yapılıp hafızadaki bir sayı hex olarak ekrana yazdırılmış oluyor.

```
key_table       .text "0123456789abcdef"            ;basılması gereken tuş tablosu
val_table       .byte 0,1,2,3,4,5,6,7,8,9           ;tuşlara göre sayısal değerler
                .byte 10,11,12,13,14,15
line            .byte 16                            ;ekrana basılacak satır sayısı
text1           .text "   c64 turkiye hex-ascii dump program"
                .byte $0d
                .text " 10-may-2005    code: hades of ascraeus"
                .byte $0d,$0d
                .text "f1:next    f3:prev.   f5:new     f7:exit"
                .byte 13,13
text2           .text " adres"
                .byte 0
text3           .text " hexadecimal memory dump  ascii "
                .byte $0d
                .text " ===== ======================= ========"
                .byte $0d,0
temp_scr        .byte 0,0,0,0,0,0,0,0               ;geçici saklama bölgesi
                .end
```





İNTRO EFEKTLERİ
1x1 TEXT FADER

```
;=============================================================================================;
; 1x1 text fade intro efekt programı                                                          ;
; C64 TÜRKİYE dergisi sayı 8 için hazırlanmıştır                                              ;
; 15-05-2005 / ismail "hades" şahin                                                           ;
; hades@amigaturk.com   www.c64turkiye.com  www.amigaturk.com  c64turkiye@amigaturk.com       ;
;=============================================================================================;

                *=$0801

prg             .word nextline
                .word 2005
                .byte $9e
                .text "2061"
                .byte 0
nextline        .word 0
```

start etiketli yere kadar olan kısımda LIST komutuyla göreceğiniz 2005 SYS2061 satırını oluşturuyoruz.

```
start           sei
                lda   #<irq
                ldx   #>irq
                ldy   #$01
                sta   $0314
                stx   $0315
                sty   $d01a
```

İnterruptları engelledikten sonra, interrupt vektörlerini bizim rutinimizi gösterecek şekilde değiştiriyoruz ve satır tarama interruptı yapacağımızı VIC'e bildiriyoruz.

```
                lda   color
                jsr   $e536
                stx   $d015
                stx   $d01d
                dex
                stx   $d020
                stx   $d021
                stx   $d027
```

Ekranı temizliyoruz ve geri dönüşte X registerinin değerini (X registerinde 1 değeri var) 1 numaralı sprite'ı açmak ($D015) ve X ekseninde genişletmek ($D01D) için kullanıyoruz. Sonra X registerini 1 azaltıp ekran renklerini ve 1 numaralı sprite'in rengini siyah yapıyoruz.

```
                lda   #$ff
loop0           sta   sdata,x
                inx
                bne   loop0
```

Sprite şekil datası için kullanılacak hafıza bölgesini $FF ile dolduruyoruz. Böylece programda kullanacağımız sprite'ın şekli dikdörtgen oluyor.

```
                lda   #(sdata/64)
                sta   $07f8
```

Sprite şekil datasının bulunduğu bölgeyi sprite pointerine yazıyoruz.

```
                ldx   #$18
                ldy   #$25
                stx   $d000
                sty   $d001
```

Sprite'ın X ve Y koordinatlarını belirliyoruz.

```
                lda   #<text1
                ldx   #>text1
                sta   get_tx+1
                stx   get_tx+2
```

Ekrana yazdıracağımız yazının başlangıç adresini programdaki ilgili yere kopyalıyoruz.

```
                ldx   #$00
loop1           lda   text1,x
                sta   $0400+13,x
                inx
                cpx   #$0e
                bne   loop1
```

İlk yazımızı ekrana yazıyoruz.

```
                lda   max_del
                sta   delay_cnt
```

Sprite hareket gecikme sayacını ayarlıyoruz

```
                lda   max_move
                sta   move_cnt
```

Sprite hareket sayacını ayarlıyoruz.

```
                lda   max_text
                sta   text_cnt
```

Ekrana yazdırılacak maksimum text sayacını ayarlıyoruz.

```
                lda   #$ee
                sta   dir
```

Sprite'ın ilk hareket yönü için "INC" komut kodunu programdaki yerine yazıyoruz.

```
                cli
                jmp   *
```

İnterruptları serbest bırakarak programı sonsuz döngüye sokuyoruz.

```
irq             inc   $d019
                dec   delay_cnt
                bne   exit
```

Hareket gecikme sayacını bir azaltıyoruz. Sıfır olmadıysa programdan çıkıyoruz.

```
                lda   max_del
                sta   delay_cnt
```

Hareket gecikme sayacı sıfır olduysa, sayacı tekrar yüklüyoruz ve devam ediyoruz.

```
                dec   move_cnt
                bne   dir
                lda   max_move
                sta   move_cnt
                lda   dir
                eor   #$20
                sta   dir
```

Hareket sayacını bir azaltıyoruz ve sıfır olmadıysa "dir" yazan yere gidiyoruz. Sıfır ise hareket sayacını tekrar yüklüyoruz. "EOR" ile hareket yönünü değiştiriyoruz.

```
                cmp   #$ee
                beq   drx
```

Hareket yönünü kontrol ediyoruz. Eğer "INC" komutu ise (yani hareket soldan sağa ise) "drx" yazan yere gidiyoruz.

```
                lda   get_tx+1
                clc
                adc   #$0e
                sta   get_tx+1
                bcc   cont
                inc   get_tx+2
```

Eğer "DEC" ise (yani sağdan sola ise) ekrana basılacak yeni yazının adresini hesaplayıp programdaki yerine yazıyoruz. Hesaplama sırasında elde biti "1" olduysa yazıların alınacağı adresin yüksek baytını 1 arttırıyoruz. Değilse aşağıda devam ediyoruz.

```
cont            dec   text_cnt
                bne   get_text
                lda   max_text
                sta   text_cnt
                lda   #<text1
                ldx   #>text1
                sta   get_tx+1
                stx   get_tx+2
```

Ekranda gösterilecek yazı sayacını 1 azaltıyoruz. Sıfır olmadıysa "get_text" bölümüne gidiyoruz. Eğer sıfır olduysa Text sayacını ayarlıyoruz ve ilk yazının adresini programda ilgili yere yazıyoruz.

```
get_text        ldx   #$00
get_tx          lda   text1,x
                sta   $040d,x
                inx
                cpx   #$0e
                bne   get_tx
```

Bu kısımda ekrana yazı yazıyoruz.

```
drx             lda   col+1
                eor   color
                sta   col+1
```

Yukarıdaki komutlarla renk belleği için kullanacağımız değeri siyah/sarı şeklinde değiştiriyoruz.

```
dir             inc   $d000
                lda   $d000
                sec
                sbc   #$10
                lsr
                lsr
                lsr
```

```
                tax
col             lda   #$00
                sta   $d800,x
```

Bu son kısımda ise sprite'ın X koordinat değerini değiştiriyoruz. Sonra bu değerden $10 (16) sayısını çıkarıyoruz. 3 kez sağa kaydırıp rengini değiştireceğimiz bellek pozisyonu için sütun değerini elde ediyoruz. Bu değeri X registerine kopyalıyoruz. En son olarak LDA komutunun parametresindeki değeri o anki renk belleği pozisyonuna yazıyoruz.

```
exit            jmp   $febc
```

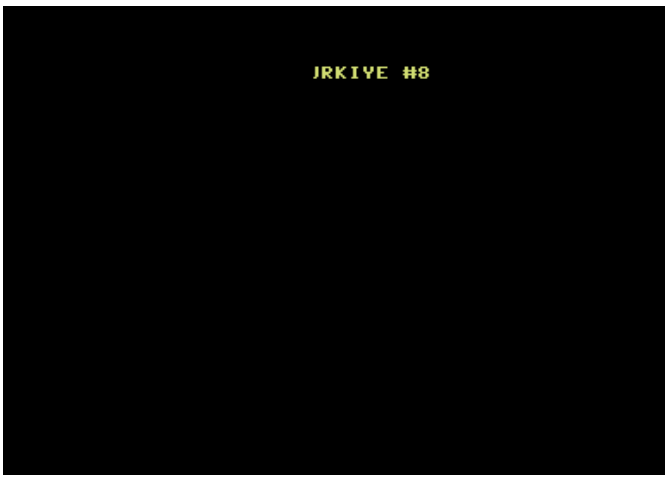
Son komutla ise kernaldaki işletim sistemi rutinine geri dönüyoruz.

```
move_cnt        .byte 0          ;O anki hareket sayacı
delay_cnt       .byte 0          ;O anki hareket gecikme sayacı
text_cnt        .byte 0          ;O anki text sayacı
color           .byte 7          ;Default renk (sarı)
max_text        .byte 10         ;Ekranda gösterilecek maksimum text sayısı
max_del         .byte $80        ;Default gecikme değeri
max_move        .byte $e0        ;Default hareket miktarı

text1           .scrl "--------------"
                .scrl "c64 turkiye #8"
                .scrl "introlar  icin"
                .scrl "sade bir efekt"
                .scrl "1x1 text fader"
                .scrl "code:hades/asc"
                .scrl "  10-05-2005  "
                .scrl "11111111111111"
                .scrl "22222222222222"
                .scrl "33333333333333"

sdata           =$09c0           ;Sprite datasının bulunduğu bölge

                .end
```



1x2 KARAKTER SETİ

```
;=============================================================================================;
; C64 için yeni karakter seti #7    (1x2 FONT)                                                ;
; C64 TÜRKİYE dergisi sayı 8 için hazırlanmıştır                                              ;
; 01-06-2005 / ismail "hades" şahin                                                           ;
; hades@amigaturk.com  www.c64turkiye.com  www.amigaturk.com  c64turkiye@amigaturk.com        ;
;=============================================================================================;
```

Yazılım ile karakter setini değiştirmeye devam ediyoruz.

```
                *=$0801

                .word nextline
                .word 2005        ; 2005
                .byte $9e         ; SYS
                .text "2061"      ; 2061
                .byte 0           ; komutu
nextline        .word 0
```

start etiketli yere kadar olan kısımda LIST komutuyla göreceğiniz 2005 SYS2061 satırını oluşturuyoruz.

```
start           sei
                lda   $01
                and   #$fb
                sta   $01
```

Önce interrupları durduruyoruz ve karakter romunu erişilebilir hale getiriyoruz.

```
                lda   #$d0
                ldx   #$30
                ldy   #$00
                sty   $fb
                sta   $fc
                sty   $fd
                stx   $fe
```

Karakter romunun ve Ram'e kopyalayacağımız yerin adreslerini tanımlıyoruz.

```
                ldx   #$10
transfer        lda   ($fb),y
                sta   ($fd),y
                iny
                bne   transfer
                inc   $fc
                inc   $fe
                dex
                bne   transfer
                lda   $01
                ora   #$04
                sta   $01
                cli
```

Yukarıdaki kısımda karakter romunu ram'e kopyalıyoruz.Daha sonra c64'ün bellek konfigürasyonunu eski haline getirip interruptları serbest bırakıyoruz.

```
new_fonts       lda   #$30
                ldx   #$34
                ldy   #$00
                sty   $fb
                sta   $fc
```

```
                sty   $fd
                stx   $fe
                lda   #$80
                sta   char_cnt
```

Bu bölümde yine adres ayarlamaları yapılıyor ve karakter sayısı belirleniyor.

```
loop2           ldy   #$00
loop0           lda   ($fb),y
                sta   temp,y
                iny
                cpy   #$08
                bne   loop0
```

Y registerini bayt sayacı olarak kullanmak üzere sıfırlıyoruz ve Ramdaki karakter setimizdeki o anki karaktere ait 8 baytı okuyup geçici olarak saklıyoruz.

```
                ldx   #$00
                ldy   #$00
loop1           lda   temp,x
                sta   ($fb),y
                iny
                sta   ($fb),y
                dey
                lda   temp+4,x
                sta   ($fd),y
                iny
                sta   ($fd),y
                iny
                inx
                cpx   #$04
                bne   loop1
```

Normalde C64'ün karakter setinde bir karakterin hem normal hem de negatif şekilleri vardır. Daha önce karakter setimizi $3000-$4000 arasına taşımıştık. Normalde C64'te 2 adet karakter seti vardır ve setler arası geçiş "SHIFT-COMMODORE" tuşlarıyla yapılır. Rom'daki birinci set $D000-$D7FF, ikinci set $D800-DFFF arasındadır. Buna göre Ram'da $3000-37FF adresleri arasında 1. set, $3800-3FFF arasında 2. set bulunacaktır. Bizim programımız 1. set ile işlem yapacaktır. $3000-$33FF arasında karakterlerin normal şekli, $3400-$37FF arasında ise negatif şekilleri vardır. Bir karakterin normal şeklinin ekran kodu 0-127 arasındadır, bu kodların 128 fazlası ise karakterin negatif şeklinin kodudur. Bizim programımız bir karakteri uzatırken hem normal, hem de negatif şekil bilgilerinin bulunduğu bölgede işlem yapmaktadır. Bu kısa bilgiden sonra işlemleri anlatalım. İlk olarak X ve Y registerlerini sıfırlıyoruz. Sonra ise daha önceden geçici olarak sakladığımız ilk baytı okuyup ilk karakter için $3000 ve $3001 adreslerine yazıyoruz. Daha sonra geçici olarak sakladığımız baytlardan 4. baytı alıp $3400 ve $3401 adreslerine yazıyoruz. Bu işlemleri 4 kez tekrarlayınca ilk karakterimizin boyunu uzatmış oluyoruz. Daha doğrusu 1x1'den 1x2'ye dönüştürmüş oluyoruz.

```
                lda   $fb
                clc
                adc   #$08
                sta   $fb
                sta   $fd
                bcc   no_hi
                inc   $fc
                inc   $fe
no_hi           dec   char_cnt
                bne   loop2
```

Bir sonraki karakterin boyunu uzatmak için gerekli hesaplamaları yapıyoruz. Karakter sayacını 1 azaltıyoruz ve sıfır oluncaya kadar yukarıdaki işlemleri tekrarlıyoruz.

```
                lda   #$01
                jsr   $e536
```

Bütün işlemler bittikten sonra yazı rengi beyaz olacak şekilde ekranı siliyoruz.

```
                lda   #$1c
                sta   $d018
```

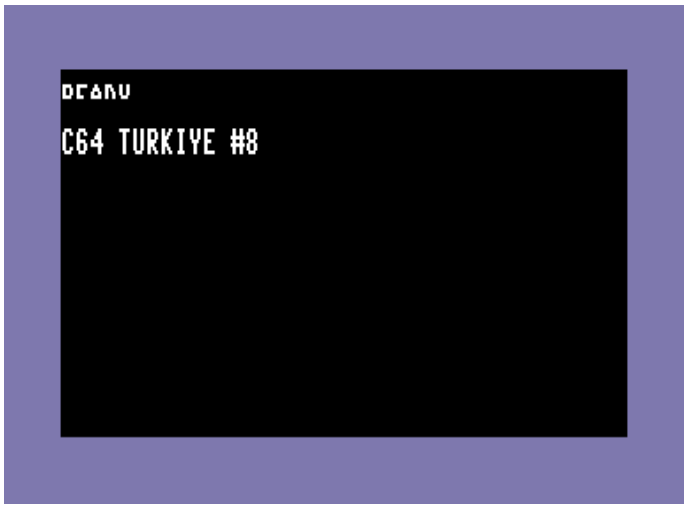
Yeni karakter setini aktif hale getiriyoruz.

```
                ldx   #$00
                stx   $d021
loop3           lda   text,x
                sta   $04a0,x
                clc
                adc   #$80
                 sta   $04a0+$28,x
                inx
                cpx   #$0e
                bne   loop3
                rts
```

Son olarak ekran koduyla bir yazı yazıyoruz. İlk satıra yazdığımız bir karakteri $80 ile toplayıp bir alt satıra yazıyoruz.

```
text            .scrl "c64 turkiye #8"        ;deneme yazısı
char_cnt        .byte 0                       ;karakter sayacı
temp            .byte 0                       ;geçici saklama yeri

                .end
```

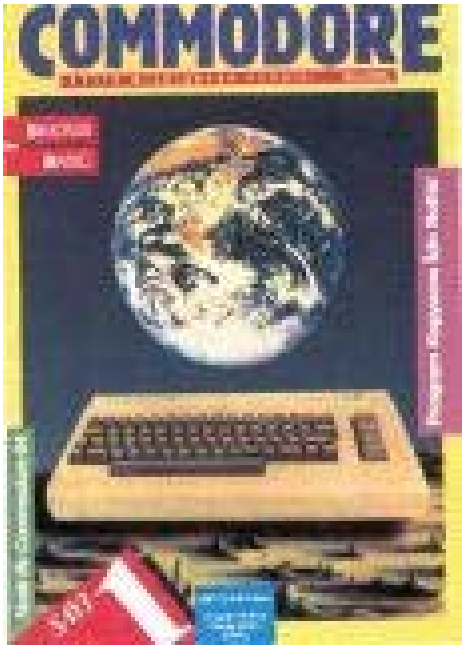


DÜNDEN BUGÜNE ÜLKEMİZDE ÇIKAN C-64 DERGİLERİ

HAZIRLAYAN : Burak ÇETİNDAĞ
ILETISIM : criminal@amigaturk.com

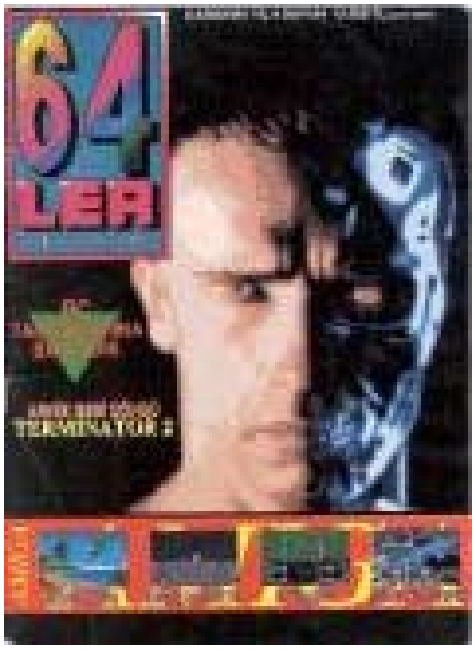
" İÇİNİZDEKİ DAHİYİ UYANDIRIN "

(Yukarıdaki slogan bir zamanlar Commodore 64'ü ülkemizde tanıtmak için kullanılırdı.)



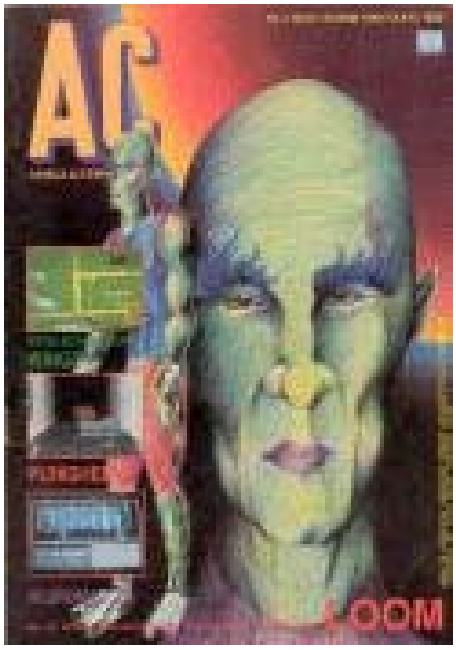
COMMODORE

Basım Yeri : Türkiye
Çıkış Tarihi : Mart 1986
Sayı : 79
Kapanış tarihi : Ekim 1992



64'ler

Basım Yeri : Türkiye
Çıkış Tarihi : Nisan 1988
Sayı : 52
Kapanış Tarihi : Haziran 1992



AC-Amiga Commodore

Basım Yeri : Türkiye
Çıkış Tarihi : Kasım 1990
Sayı : 2
Kapanış Tarihi : Aralık 1990

C64 TÜRKİYE

Basım yeri : Türkiye (E-dergi)
Çıkış Tarihi : Kasım 2002
Sayı : 8
Kapanış Tarihi : Ekim 2005

PROGRAM DÖKÜMLERİ

PROGRAM ADI: MEMDUMP 0801 0A17

```
0801: 0B 08 D5 07 9E 32 30 36  - 43BF
0809: 31 00 00 00 A9 0D 20 36  - 2B2B
0811: E5 CA 8E 20 D0 8E 21 D0  - 9184
0819: 86 FB 86 FC A9 45 A0 09  - 7E40
0821: 20 1E AB A9 3A 20 D2 FF  - 8EE9
0829: A9 24 20 D2 FF 20 D4 08  - 6FDC
0831: 85 FC 20 D4 08 85 FB A0  - 97B1
0839: 00 A2 06 18 20 F0 FF A9  - 85A2
0841: BE A0 09 20 1E AB A9 C5  - 81BE
0849: A0 09 20 1E AB A9 10 8D  - 61C2
0851: 44 09 A9 20 20 D2 FF A5  - 8916
0859: FC 20 11 09 A5 FB 20 11  - 56F5
0861: 09 A9 3A 20 D2 FF A9 20  - 76B4
0869: 20 D2 FF A0 00 B1 FB 99  - 9FCA
0871: 0F 0A 20 11 09 A9 20 20  - 2C8E
0879: D2 FF C8 C0 08 D0 EE A0  - B29D
0881: 00 B9 0F 0A C9 20 90 05  - 4564
0889: C9 60 B0 01 2C A9 2E 20  - 50E5
0891: D2 FF C8 C0 08 D0 EA A9  - B40C
0899: 0D 20 D2 FF A5 FB 18 69  - 874B
08A1: 08 85 FB 90 02 E6 FC CE  - AA96
08A9: 44 09 D0 A6 20 E4 FF C9  - A6AB
08B1: 85 F0 12 C9 86 F0 0F C9  - 96AC
08B9: 87 F0 07 C9 88 D0 ED 4C  - 982E
08C1: 74 A4 4C 0D 08 2C C6 FC  - 7D4B
08C9: A0 00 A2 08 18 20 F0 FF  - 82D3
08D1: 4C 4E 08 20 EB 08 20 01  - 3226
08D9: 09 0A 0A 0A 0A 85 FB 20  - 4761
08E1: EB 08 20 01 09 05 FB 85  - 5BDA
08E9: FB 60 20 E4 FF A2 00 DD  - 9DB9
08F1: 24 09 D0 04 20 D2 FF 60  - 7764
08F9: E8 E0 10 D0 F2 4C EB 08  - 88CD
0901: A2 00 DD 24 09 F0 05 E8  - 7CBD
0909: E0 10 D0 F6 BD 34 09 60  - 7496
0911: 48 4A 4A 4A 4A 20 1C 09  - 2D73
0919: 68 29 0F AA BD 24 09 20  - 4376
0921: D2 FF 60 30 31 32 33 34  - 503F
0929: 35 36 37 38 39 41 42 43  - 3D1F
0931: 44 45 46 00 01 02 03 04  - 11BD
0939: 05 06 07 08 09 0A 0B 0C  - 0960
0941: 0D 0E 0F 00 20 20 20 43  - 2051
0949: 36 34 20 54 55 52 4B 49  - 46D1
0951: 59 45 20 48 45 58 2D 41  - 4153
0959: 53 43 49 49 20 44 55 4D  - 4660
0961: 50 20 50 52 4F 47 52 41  - 481B
0969: 4D 0D 20 31 30 2D 4D 41  - 3622
0971: 59 2D 32 30 30 35 20 20  - 2D27
0979: 20 20 43 4F 44 45 3A 20  - 3657
0981: 48 41 44 45 53 20 4F 46  - 42FE
0989: 20 41 53 43 52 41 45 55  - 47F6
0991: 53 0D 0D 46 31 3A 4E 45  - 3A49
0999: 58 54 20 20 20 20 46 33  - 320B
09A1: 3A 50 52 45 56 2E 20 20  - 3689
09A9: 20 46 35 3A 4E 45 57 20  - 3B61
09B1: 20 20 20 20 46 37 3A 45  - 349E
09B9: 58 49 54 0D 0D 20 41 44  - 3490
09C1: 52 45 53 00 20 48 45 58  - 3FA3
09C9: 41 44 45 43 49 4D 41 4C  - 470C
09D1: 20 4D 45 4D 4F 52 59 20  - 41CF
09D9: 44 55 4D 50 20 20 41 53  - 40D8
09E1: 43 49 49 20 0D 20 3D 3D  - 3214
09E9: 3D 3D 3D 20 3D 3D 3D 3D  - 39B7
09F1: 3D 3D 3D 3D 3D 3D 3D 3D  - 3D00
09F9: 3D 3D 3D 3D 3D 3D 3D 3D  - 3D00
0A01: 3D 3D 3D 20 3D 3D 3D 3D  - 39B7
0A09: 3D 3D 3D 3D 0D 00 00 00  - 168B
0A11: 00 00 00 00 00 00 00 00  - 0000
```

PROGRAM ADI: KARSET7 0801 08B8

```
0801: 0B 08 D5 07 9E 32 30 36  - 43BF
0809: 31 00 00 00 78 A5 01 29  - 333E
0811: FB 85 01 A9 D0 A2 30 A0  - 8AF0
0819: 00 84 FB 85 FC 84 FD 86  - A869
0821: FE A2 10 B1 FB 91 FD C8  - BC58
0829: D0 F9 E6 FC E6 FE CA D0  - E25B
0831: F2 A5 01 09 04 85 01 58  - 4593
0839: A9 30 A2 34 A0 00 84 FB  - 8598
0841: 85 FC 84 FD 86 FE A9 80  - B18F
0849: 8D B6 08 A0 00 B1 FB 99  - 8EC2
0851: B7 08 C8 C0 08 D0 F6 A2  - A03B
0859: 00 A0 00 BD B7 08 91 FB  - 89AC
0861: C8 91 FB 88 BD BB 08 91  - 929D
0869: FD C8 91 FD C8 E8 E0 04  - A7E1
0871: D0 E9 A5 FB 18 69 08 85  - 7C61
0879: FB 85 FD 90 04 E6 FC E6  - C0AD
0881: FE CE B6 08 D0 C5 A9 01  - 825F
0889: 20 36 E5 A9 1C 8D 18 D0  - 7967
0891: A2 00 8E 21 D0 BD A8 08  - 6C82
0899: 9D A0 04 18 69 80 9D C8  - 7FD5
08A1: 04 E8 E0 0E D0 EF 60 03  - 72FE
08A9: 36 34 20 14 15 12 0B 09  - 1591
08B1: 19 05 20 23 38 00 00 09  - 11BA
```

```
PROGRAM ADI: FADER                0801 0982

0801: 0B 08 D5 07 9E 32 30 36  - 43BF
0809: 31 00 00 00 78 A9 7C A2  - 6368
0811: 08 A0 01 8D 14 03 8E 15  - 3AE8
0819: 03 8C 1A D0 AD F2 08 20  - 6512
0821: 36 E5 8E 15 D0 8E 1D D0  - 8603
0829: CA 8E 20 D0 8E 21 D0 8E  - 8A65
0831: 27 D0 A9 FF 9D C0 09 E8  - A3C5
0839: D0 FA A9 27 8D F8 07 A2  - 8F58
0841: 18 A0 25 8E 00 D0 8C 01  - 5598
0849: D0 A9 F6 A2 08 8D C7 08  - 79DF
0851: 8E C8 08 A2 00 BD F6 08  - 6FE7
0859: 9D 0D 04 E8 E0 0E D0 F5  - 9CF5
0861: AD F4 08 8D F0 08 AD F5  - A0AE
0869: 08 8D EF 08 AD F3 08 8D  - 7B39
0871: F1 08 A9 EE 8D DA 08 58  - 80E3
0879: 4C 79 08 EE 19 D0 CE F0  - A31A
0881: 08 D0 68 AD F4 08 8D F0  - 9982
0889: 08 CE EF 08 D0 4B AD F5  - 9E66
0891: 08 8D EF 08 AD DA 08 49  - 67F0
0899: 20 8D DA 08 C9 EE F0 30  - 8D52
08A1: AD C7 08 18 69 0E 8D C7  - 70E1
08A9: 08 90 03 EE C8 08 CE F1  - 985C
08B1: 08 D0 10 AD F3 08 8D F1  - 91B6
08B9: 08 A9 F6 A2 08 8D C7 08  - 6C97
08C1: 8E C8 08 A2 00 BD F6 08  - 6FE7
08C9: 9D 0D 04 E8 E0 0E D0 F5  - 9CF5
08D1: AD E8 08 4D F2 08 8D E8  - 90A9
08D9: 08 EE 00 D0 AD 00 D0 38  - 6CED
08E1: E9 10 4A 4A 4A AA A9 00  - 5C4A
08E9: 9D 00 D8 4C BC FE 00 00  - 61EB
08F1: 00 07 0A F0 E0 2D 2D 2D  - 4FD0
08F9: 2D 2D 2D 2D 2D 2D 2D 2D  - 2D00
0901: 2D 2D 2D 03 36 34 20 14  - 2280
0909: 15 12 0B 09 19 05 20 23  - 15AE
0911: 38 09 0E 14 12 0F 0C 01  - 0E69
0919: 12 20 20 09 03 09 0E 13  - 0FC0
0921: 01 04 05 20 02 09 12 20  - 103D
0929: 05 06 05 0B 14 31 18 31  - 1B1D
0931: 20 14 05 18 14 20 06 01  - 0F04
0939: 04 05 12 03 0F 04 05 3A  - 1328
0941: 08 01 04 05 13 2F 01 13  - 0F2A
0949: 03 20 20 31 30 2D 30 35  - 2B38
0951: 2D 32 30 30 35 20 20 31  - 2BE9
0959: 31 31 31 31 31 31 31 31  - 3100
0961: 31 31 31 31 31 32 32 32  - 3189
0969: 32 32 32 32 32 32 32 32  - 3200
0971: 32 32 32 33 33 33 33 33  - 32C5
0979: 33 33 33 33 33 33 33 33  - 3300
0981: 33 00 00 00 00 00 00 00  - 0363
```

KATKILARINDAN DOLAYI

DENİZ CAN ÇELİK'e
BEKİR "SLOWHAND" OGURLU'ya
BİLGEM "NIGHTLORD" ÇAKIR'a
BURAK "CRIMINAL-FX" ÇETİNDAĞ'a

TEŞEKKÜR EDERİM.

İSMAİL "HADES" ŞAHİN