C64 TÜRKİYE

SAYI : #$03 EYLÜL 2003

BİLGİ PAYLAŞTIKÇA ARTAR

C64'ÜN YENİ SAHİBİ:

IRONSTONE PARTNERS

AYRICA...

- 6510 ASSEMBLER KURSU – 3
- HARDSID QUATTRO PCI
- C64ASM V1.1
- TEXT ŞİFRE/DEŞİFRE PROGRAMI
- AEGIS/BRONX RÖPORTAJI
- …VEEE TABİİ Kİ COMMODORE-ONE!

C64 TÜRKİYE

SAYI : #$03 **EYLÜL 2003**

BİLGİ PAYLAŞTIKÇA ARTAR

Sahibi ve Büyük Patron : İsmail **"HADES"** Şahin
Haberler ve Çeviriler : Deniz Can Çelik
Kapak Resmi : Yeni C64 Logoları

İletişim: hades6510@yahoo.com

İÇİNDEKİLER

M A N İ F E S T O:

1- ***Dergi için COMMODORE 64 ile ilgili her türlü yazı, resim, program, oyun tanıtımı, ipuçları, hile ve ufak-tefek rutinler gibi malzemeler kabul edilir.***
2- ***Yukarıdaki malzemelerin çalışır durumda olduğundan emin olunmalıdır.***
3- ***Gönderilen malzemeler dergide mutlaka yayınlanacak diye herhangi bir şart yoktur.***
4- ***Malzemeyi gönderenin adı-soyadı-e-mail adresi-oturduğu şehir ve kendisini tanıtan kısa bir not yazması tercih edilir.***
5- ***Derginin editörü gönderilen malzemelerde değişiklik, düzenleme vs yapabilir.***
6- ***Gönderilen malzemelerin sorumluluğu gönderene aittir.***
7- ***Hiç bir şekilde siyasi, dini yazılara yer verilmeyecektir. Ayrıca kişilere ve şirketlere yönelik hakaret, aşağılama, küfür gibi yazılar yayınlanmayacaktır.***
8- ***Emeğe saygı gereği kaynak göstermek şartıyla herkes dergideki yazılardan alıntı yapabilir.***
9- ***BU DERGİ KESİNLİKLE TİCARİ AMAÇLI DEĞİLDİR VE PARA İLE SATILAMAZ.***

EDİTÖRÜN KLAVYESİ

ÖNSÖZ : Herkese merhaba... Bu sayıyı bilerek geciktirdik çünkü Tulip firmasının yaptığı bir açıklama sonucu C64 dünyası birden hareketlendi.

Konuyla ilgili yazıları derginizin ilerleyen sayfalarında görebilirsiniz. Bu konuda benim görüşlerim ise şöyle: PC'lerin her geçen gün daha fazla para canavarına döndüğü günümüzde 20 sene öncesinin bir ev bilgisayarına isim hakkı olarak bir çuval para sayan bir şirket acaba C64 kullanıcılarını yolunacak kaz olarak mı görüyor ? Eğer böyleyse eline hiçbir şey geçmeyecektir. Kendi hesaplarına göre 6 milyon C64 kullanıcısı var ve kulağa oldukça hoş geliyor. Her kullanıcıdan 1$ gelir olsa 6 milyon $ eder. İyi para! Fakat C64'çülerin gösterdiği tepkiyi gören Ironstone yöneticileri 180° dönüş yaparak C64 için yeni ürünler çıkartmak vs... demeye başlamıştır. İşte burada durup sormak lazım, bunca zamandır neredesiniz diye. Bence, C-1'in doğumu yaklaştıkça, "elimizde isim hakkı var, o halde bu durumdan kendimize nasıl pay çıkarabiliriz" derdine düşmüşlerdir. Eğer gerçekten C64 için yeni donanım geliştirmek gibi bir niyetleri olduysa, bunu 3-5 sene önce de yapabilirlerdi. Sinekten yağ çıkarma misali. Gelişmeleri sizlere duyurmaya devam edeceğiz....

Bu konuyu bir kenara bırakalım ve devam edelim. Assembler kursu bu sayıda sona eriyor. Umarım işinize yaramıştır. Komutların ne işe yaradığını bilmek yeterli değildir. Bir program yazmaya başlayınca komutların nasıl ve nerede kullanılacağını da bilmek gerekir. Bu nedenle bir assembler program yazmaya başlamalısınız. PC üzerinde bir C64 assembler programı yazabilmek için elinizde olması gerekenler şunlardır. WORDPAD, C64ASM v1.1 ve son olarak yazdığınız programı çalıştırmak için bir C64 emülatörü. Bu sayıda C64ASM programının açıklamasını bulabilirsiniz. Son olarak bir C64'çü ile yapılan bir röportajımız var.

Bir sonraki sayıda görüşebilmek dileğiyle...

Jeri Ellsworth "mesajı" veriyor: C64 RULES!

HABERLER

BRONX 7D3 DEMO PARTY

Türkiye'deki C64 scene'inin en eski ve önde gelen grubu Bronx, 23 ve 24 ağustos tarihlerinde İstanbul Kadıköy'de 7D3 Demo Partisini düzenledi. Partiye C64 alanında katılan demoları sizlere dördüncü sayımızda tanıtacağız. Partiyle ilgili haberler ve fotoğraflar için Bronx'un internet sitesine gözatabilirsiniz.
http://www.bronxwhq.org/

COMMODORE TÜRK FORUMU AÇILDI

Türkiye'deki Commodore kullanıcılarının yeni forumu Amigaturk.net bünyesinde açıldı. Bu ay röportaj da yaptığımız Aegis/Bronx tarafından açılan forumda Commodore bilgisayarlarıyla ilgili genel tartışma bölümünün dışında yazılım/donanım ve alım-satım bölümleri de bulunuyor. Aegis'i böyle bir forum açtığı için tebrik ediyoruz.
http://www.amigaturk.net/forum

LOTEK64, HADES'LE RÖPORTAJ YAPTI

Alman C64 dergisi Lotek64, 2. sayımızın ardından Hades ile bir röportaj yaptı. Türkiye'de C64 kullanımı ve C64 scene'i üzerine olan bu röportaj, derginin 7. sayısında yayınlanacak. Lotek64'ün 6. sayısını aşağıdaki adresten indirebilirsiniz:
http://www.galaktus.de/downloads/lotek/Main/download/Lotek64_06.zip
Lotek64 web sitesi: http://www.lotek64.com/

BACK IN TIME LIVE 4!

C64 rock müzik festivali Back In Time'ın (BIT) 4.'sü 13 eylül Cumartesi günü Brighton/İngiltere'de yapılacak. "Dünyanın en büyük 8 bit rock konseri" diye de adlandırılan BIT4'ün en özel konuğu ise festivalin odağını oluşturan SID çipinin yaratıcısı Martin Galway! Festivalin önde gelen katılımcıları ise 80'lerin oyun müziklerinin kralı olarak adlandırılan Rob Hubbard ve Danimarkalı grup Press Play on Tape. Biletler ve konser hakkındaki tüm detaylar için BIT web sayfasına gözatabilirsiniz: www.backintimelive.com. Konsere katılan grupların albümlerini ise www.c64audio.com sitesinde bulabilirsiniz.

SCENE WORLD #8 ÇIKTI

3 ayda bir yayınlanan C64 disk dergisi Scene World, 8. sayısını ağustos ayı içinde çıkardı. Bu sayı aynı zamanda, içerdiği 2000 block'tan fazla yazı ile, Scene World'ün çıkardığı en büyük sayı ünvanını taşıyor. Dergi .d64 formatında olduğu için çalıştırmak için bir emülatör gerekiyor. Dergiyi aşağıdaki adresten indirebilirsiniz:
http://home.arcor.de/sceneworld/Sceneworld2/Swo/downloads/issues/swo_08.zip

VICE 1.12 ÇIKTI

En iyi C64 emülatörlerinden VICE'ın 1.12 sürümü çıktı. VICE emülatörü Windows'un dışında RISC OS, OS/2, BeOS ve Mac OSX'i de destekliyor.
http://viceteam.bei.t-online.de/#download

SONY-ERICSSON P800 İÇİN C64 EMÜLATÖRÜ: FRODO!

Commodore 64'ün cep telefonlarına girmesi yeni birşey değil aslında. Ama eğer bir SonyEricsson P800'ünüz varsa Frodo'yu kullanabilirsiniz. 108 KB'lık bu program tamamen C++ ile yazılmış ve freeware. Frodo emülatörü .D64 dosyalarını çalıştırabiliyor.
http://my-symbian.com/uiq/download/sendfile.php4?DownloadID=13

COMMODOREONE GÜNLÜĞÜ

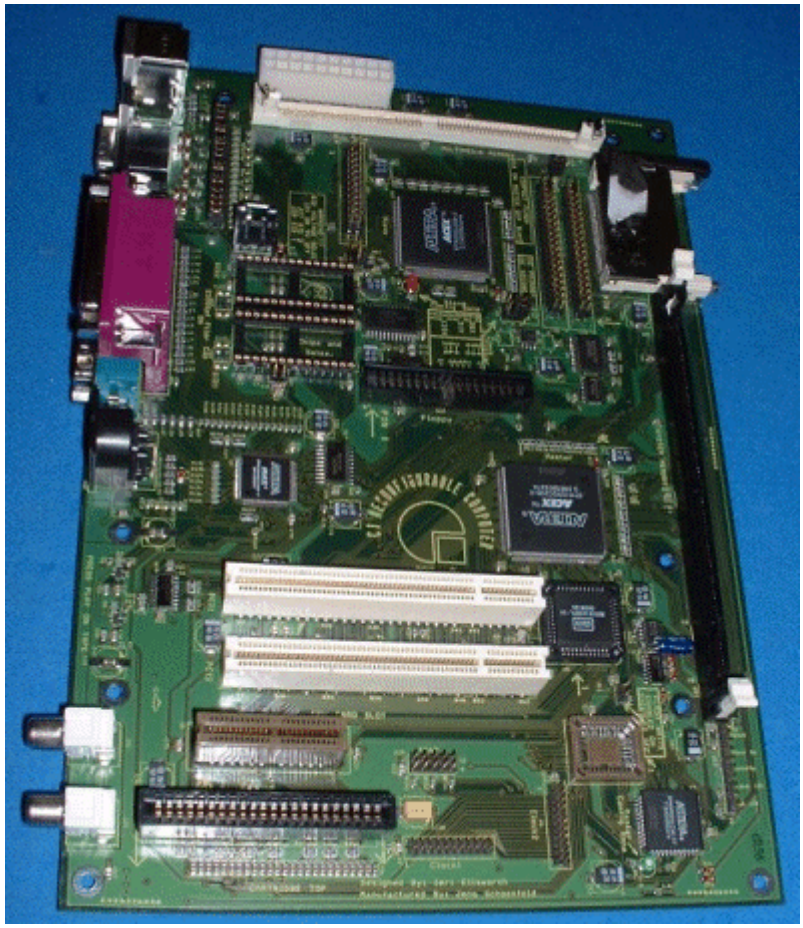
10 NİSAN – 2. sayımızı çıkardıktan kısa bir süre sonra 10 nisanda C-One üretimi başladı. Yapılan açıklamada kartların 5 mayısta Almanya ve Hollanda'da, 9 mayısta da Avrupa'nın geri kalanı ve Kuzey Amerika'da satışa çıkacağı söylendi.

13 NİSAN – C-One'ın sistem veriyolu frekansı 100 Mhz'i aştı! İlk prototipte 50 MHz'lik bir frekansta çalışan veriyolu, rev. 2 versiyonunda 33.87 MHz'den 135.48 MHz'e kadar seçilebilecek bir yelpazeye ulaştı. Böylece kullanıcılar FSB frekansını ayarlayabilecek hale geldiler. Ancak kartların bütün SIMM modülleriyle uyumluluk göstermesi amacıyla, sistem veriyolunun 67.74 MHz'de sabitlenmesine karar verildi.

4 MAYIS – C-One'ın 4 mayısta yapılacak olan C1 Release Party'de satılmaya başlanması planlanıyordu. Ancak early startup ROM'larının güncellenmesi sebebiyle bu gerçekleşmedi. Bunun yerine, üretilen prototip model basına tanıtıldı.

26-27 HAZİRAN – C-One projesinin yaratıcısı Jeri Ellsworth, projenin finansörü ve üreticisi Jens Schönfeld (Individual Computers) ile, haziran ayında Sacramento/California'daki AmiWest 2003 fuarına katılarak prototipi bir kez daha sergilediler.

22-27 TEMMUZ – Jeri Ellsworth, Kansas'taki Apple II KFest'e katıldı. Burada Apple II kullanıcılarına C-One'ı anlatan Jeri, ayrıca Apple II programcılarını Commodore tarafına çekmeye çalıştı!



İŞTE C-ONE PROJESİNİN SON HALİ! (REV 2 ANAKART)

C64'ÜN YENİ SAHİBİ: IRONSTONE PARTNERS!

Geçtiğimiz temmuz ayında Commodore dünyası son zamanların en olaylı günlerini yaşadı. 1997'den beri Commodore'un isim haklarını elinde bulunduran Hollandalı Tulip firması, bu hakların kullanımı konusunda Ironstone Partners şirketiyle bir anlaşma imzaladı. Bu olay o kadar büyük bir yankı yarattı ki, dergiyi temmuz ayında çıkartmayı planladığımız halde, gelecek haberleri izlemek amacıyla bir süre daha beklemeye karar verdik.

TULIP'İN İLK AÇIKLAMASI

11 temmuz günü C64 kullanıcılarının uzun süren sakinliği Hollanda'dan gelen bir haberle bozuldu. Tulip tarafından yapılan açıklamada, şirketin, Ironstone tarafından pazarlanan her C64 ürününden bir lisans ücreti alacağı belirtildi. Yapılan resmi açıklama şöyleydi:
"Tulip, Ironstone tarafından pazarlanan her C64 ürününden (emülatör veya başka yazılımlar ve donanımlar), ayrıca download'lardan ve reklamlardan bir lisans ücreti alacaktır.
Bugün resmi olmayan internet sitelerinde, 6 milyona yakın Commodore kullanıcısı bulunmaktadır. Tulip bu anlaşmayla Ironstone'a resmi C64 portalı kurma ve Commodore C64 isimlerini kullanma yetkilerini vermiştir.
Tulip'ten izinsiz olarak Commodore veya C64 isimlerini kullanan yaklaşık 300 ticari site vardır. Tulip, Commodore isminin izinsiz kullanımına izin vermeyecektir. Ironstone ve Tulip, Commodore kullanıcılarını resmi C64 portalına katılmaya davet etmektedir.
Bu ortaklık sonucu, Ironstone, resmi C64 oyunlarını ve portalını, bir ücret karşılığında kullanıcıların hizmetine sunacaktır. Bu portalın amacı, devasa sayıdaki C64 kullanıcısı birleştirmek, yeni resmi C64 emülatörünü farklı yazılım ve donanım formatlarına uyumlu olarak piyasaya sunmak, yeni oyunların yanında ünlü klasikleri ve Commodore markalı ürünleri satışa sunmaktır.
Tulip, bu portal sayesinde, ilerleyen aylarda resmi C64 emülatörünü kullanabilen, Commodore markalı yeni donanım ürünlerini piyasaya sunacaktır."

Bu açıklama C64 kullanıcıları arasında büyük yankı uyandırdı. Hatta Türkiye'deki günlük gazetelerin haber sitelerinde bile kendine geniş yer buldu. Kullanıcıların çoğu, özellikle internet sitelerinin kapatılması konusuna büyük tepki gösterdi. Ayrıca 6 milyon kullanıcı ve 300 kadar ticari site olduğu savı da bayağı bir tartışıldı. (Burada bir görüş belirtmekte yarar var: Tulip'in açıklamasının daha işin başında C64 kullanıcılarının sert tepkisiyle karşılaşmasının, Tulip'in insanları hiçe sayarak kesin çizgiler çekmesinden kaynaklandığını düşünüyorum. Bundan başka, Commodore adının scene felsefesine ters bir biçimde ve en önemlisi de C64 scene'ini karşısına alarak ticari amaç için kulanılması, insanlarda ister istemez antipati uyandırdı. Bugün C64'e sahip çıkan kullanıcılar, Tulip'in ortalarda gözükmediği bunca yıl boyunca karşılık beklemeden C64'ü yaşatmış ve C64 ruhunun yaşamasına katkıda bulunmuşlardır.)

C64'ÜN YENİ KORUYUCU MELEĞİ

İlk açıklamadan sonra kullanıcı gruplarının gösterdiği tepki ve ortalığın tam anlamıyla "birbirine girmesi" üzerine Ironstone yöneticilerinden Darren Melbourne, 15 temmuzda bir İnternet sitesine bazı açıklamalar yaptı:
"Yasal işlem yapacağımız kişiler sadece Commodore markasını suistimal edenlerdir. C64 emülatörleri ve oyunları satan büyük firmalar var. Durdurmak istediğimiz insanlar bunlar. Lemon64 gibi yaklaşık 7 yıllık siteleri kapatmak mı? Asla! Lemon64'ü son birkaç yıldır okuyorum ve harika. Fan siteleri mali açıdan problemler yaşıyor. Onlar olmasaydı C64 10 yıl önce ölmüş olurdu. Bizim görevimiz onlara yardım etmektir. Şu anda yapmak istediğimiz son şey düşman edinmek."
Melbourne ayrıca, Ironstone'un C64 logosunu değiştireceğini, ama yeni logonun eskisiyle %90 benzerlik taşıyacağını da söyledi.

Ironstone'un C64 ve Commodore'un isim haklarını alması, 16 temmuz günü Ironstone'un resmi İnternet sitesinden yayınlanan bir açıklamayla da duyuruldu.

TEPKİLER VE AÇIKLAMALAR...

Her ne kadar Ironstone'dan açıklamalar yapılsa da C64 kullanıcılarının, özellikle de Tulip'in yaptığı açıklamaya karşı tepkisi büyüktü ve bitmedi. Commodore Scene dergisinin editörü Allan Bairstow da Retro Mart sitesinde yayınlanan yazısında, yapılan basın bültenlerine tepkisini şu sözlerle gösterdi:
"Buna ne demeli: Yaklaşık 6 milyon kişiye hizmet götürmekten bahsediyorlar. Bu sayıyı da nerden bulmuşlar? Dürüst olmak gerekirse C64 scene'i aşırı derecede aktif, ama 6 milyon düzenli veya gerçek

kullanıcı olduğundan şüphe ediyorum. (...) Tulip'e bazı tavsiyelerim var çünkü belli ki ne söylediklerinden haberleri yok:

1. *C64 scene'inin sırtından kazanılacak para yok,*
2. *Scene hâlâ var çünkü maliyeti çok az,*
3. *Herhangi bir copyright yasasını uygulamaya kalkışmayın çünkü hedeflediğiniz insanları kaybetmiş olursunuz."*

Bir süre sonra Ironstone'dan Paul Gauge, Allan Bairstow'un sözlerine cevap verdi. Paul Gauge, C64'ün haklarını niye aldıklarına değindi:
"Bu kadar parayı C64'ün haklarına ne diye ödedik? Bir firma olarak, hissedarlarımıza karşı gelir oluşturmak gibi bir yükümlülüğümüz var. Fakat bunu, kullanıcıların zaten ücretsiz edindikleri şeyleri ücretli yaparak değil, bir dizi C64 donanım ve yazılım ürünlerini piyasaya sürerek gerçekleştireceğiz. Bunun hem varolan topluluk, hem de yeni nesil kullanıcıların hoşuna gideceğini düşünüyoruz."
Paul Gauge, ayrıca Ironstone'un C64 hakkındaki hedeflerini de açıkladı:
"Ironstone'un C64 için iki ana hedefi var: Yeni Commodore 64 ürünleri geliştirerek, varolan sadık Commodore 64 kullanıcılarına hizmet etmek ve Commodore 64 topluluğunu genişletmek. Bunun dışında C64 ve Commodore isimlerini tekrar gündeme getirmek ve daha da geniş bir kitleye Commodore 64 keyfini yaşatabilmek için yeni ürünler geliştirmek."

YENİ LOGOLAR

31 temmuz tarihinde ise Ironstone, merakla beklenen yeni Commodore 64 logolarını tanıttı. Logolar söylendiği gibi bazı ufak tefek değişiklikler dışında hemen hemen aynıydı. Yeni logoları bu sayımızın kapağında da görebilirsiniz.
Bu değişiklik Commodore'un iflasından sonra yapılan ilk değişiklik değil. Escom firması da Commodore'u satın aldığı 1995'te logolar üzerinde ufak bir değişiklik yapmış hatta çıkardığı bazı yeni ürünlerin üzerinde bu logoyu kullanmıştı, ama bütün bunlara rağmen bu yeni logo pek tutulmamıştı.

Escom'un Commodore logosu...

SONUÇ

Peki bundan sonra ne olacak? Kullanıcıların endişe duyduğu konulardan ilki ROM paylaşımının yasaklanması. Tulip'in yaptığı ilk açıklamada yeni ve resmi bir C64 emülatörünün yapılacağı ve satılacağı söyleniyordu. Ama kullanıcıların görüşü C64'ün çok zor emüle edilebilen bir makine olduğu ve bir emülatör yazmak için uzun zaman gerektiği yönünde. Gerçekten de VICE ve CCS64 gibi emülatörler, geçen onca zamandan sonra bugün bile hâlâ bazı aksaklıklar yaşayabiliyorlar.
İkinci bir soru işareti ise merakla beklenen CommodoreOne konusunda geliyor. Aslında Jeri Ellsworth ve Jens Schönfeld, Tulip ile önceden bir iletişim kurmuşlar. Ama Jens Schönfeld'in söylediğine göre konuşmalar daha başlamadan sona ermiş. Eğer Tulip ve Ironstone isim hakkı doğrultusunda yasal yollara başvururlarsa, CommodoreOne projesi için sıkıntı başgösterebilirdi. "Dİ" diyoruz çünkü insanların CommodoreOne demesine rağmen projenin resmi adı "C-One" (Reconfigurable Computer). Bu yüzden CommodoreOne konusunda endişelenmeye (en azından şimdilik) gerek yok. Şu anda sadece C-One'ı C64'e çevirmek için kullanılacak Compact Flash kartındaki yazılım için lisans ücreti istenebilir.
Commodore veya C64 ismini kullanan sitelerin kapatılması konusu da en çok tepki çeken konulardan biriydi. Darren Melbourne'ün Lemon64 hakkında söylediklerini yukarıda yazmıştık. Benzer bir şekilde Paul Gauge da "Ironstone'un, C64'ü ticari amaçlarla kullanan siteler hakkında yasal yollara başvurmadan önce, ortak hareket etmeyi daha uygun bulduğunu; ticari amaçlı olmayan siteleri ise kapatmak veya engellemek gibi bir niyeti olmadığını" söyledi.
Tulip ve Ironstone'un yeni Commodore 64'ler üretmesi ise şu an için uzak bir ihtimal gibi duruyor. Nedeni ise C64 ve SID/VIC çipleri gibi diğer Commodore donanımlarının patentlerinin Tulip'in değil, Gateway'in elinde bulunması. Ayrıca C64 çiplerinin sahip olduğu teknoloji çok eski olduğundan, onları tekrar üretecek sistemi kurmak çok pahalıya patlayabilir. Aynı şekilde ROM'ların hakları da Gateway'in elinde. Ancak bu konuda fazla bir bilgi olmadığından ve basın bültenlerinde veya diğer haberlerde de Gateway'in adı geçmediğinden ne olacağı şimdilik merak konusu...
Tulip'in kurduğu "Commodore Portalı" ise *www.commodore.net* adresinde açıldı. Site şu an için boş.

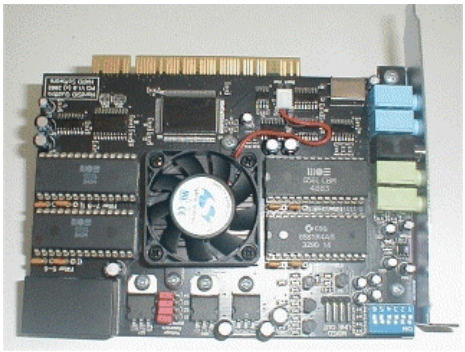
Eğer Ironstone'a görüşlerinizi bildirmek isterseniz *info@ironstonepartners.com* adresine yazabilirsiniz. Önerilerinizi de *c64ideas@ironstonepartners.com* adresine yollayabilirsiniz. Ironstone yetkilileri, ilk açıklamanın yapıldığı günden bu yana aldıkları olumlu/olumsuz tepki ve eleştirilerin sayısından oldukça memnun.

HardSID Quattro PCI

Herkese merhaba! Bu sayımızda sizlere harika bir ürün olduğunu düşündüğümüz HardSID Quattro PCI ses kartını tanıtacağız. Eğer 64'ün SID büyüsünden kurtulamadıysanız ve Windows altında da SID kullanmaya devam etmek istiyorsanız, bu kart tam size göre demektir. HSQ PCI, daha önce HardSID ISA'yı üreten firmanın çıkardığı gelişmiş bir PCI ses kartı. Ve şu anda Windows için MIDI desteğine sahip tek SID kartı olma özelliğini taşıyor. Kart Windows XP, 2000, Me, 9x uyumlu ve tüm MIDI yazılımlarıyla sorunsuz biçimde çalışmakta (Logic, Fruity Loops, Cubase, Cakewalk, vs..). Bunun için MIDI çıkışını HSQ PCI olarak göstermeniz yetiyor. MIDI yazılımlarından başka, VICE emülatörü (geçtiğimiz günlerde 1.12 versiyonu çıktı) ve SIDPlay2 de HSQ PCI'ı destekliyor. Kartın üzerine 4 taneye kadar SID çipi takabiliyorsunuz (6581 veya 8580 farketmiyor ama profesyonel kullanımlar için 8580 öneriliyor). Gerçi üretici firma tek bir SID çipinin bile yeterince güzel sonuç verdiğini söylüyor. Ama siz isteğinize göre SID'lerin sayısını arttırabilir ve 12 kanala kadar sesi kullanabilirsiniz. Ayrıca kartın üzerinde ayrı ayrı 6581 ve 8580 çiplerini aynı anda kullanmak mümkün.

HSQ PCI'ın çıkışları...

Kartın bir ses kartı olarak ilginç bir özelliği de üzerinde fan bulunması! Bu özellik SID çiplerinin aşırı ısınmasından kaynaklanıyor. 4 tane çipin birden kart üzerinde bulunması ihtimali de gözönüne alınınca bir fan koymak gerekmiş...



Şimdi kartın giriş/çıkışlarını inceleyelim. HSQ PCI'da 1 adet internal ve 5 adet external olmak üzere 6 tane konnektör var. Bunlardan internal olanı tahmin edilebileceği gibi CD-ROM'a bağlanması için. Ama eğer isterseniz başka bir ses kartına da bağlayabilirsiniz. External olanlar stereo jack şeklindeler. Aşağıdaki şemada görünüyorlar. Burada açıklanması gereken şey sanırım ortadaki siyah karışık çıkış. Eğer kartınızla kullanacağınız profesyonel aletler yoksa bu çıkış işinize yarayabilir...

Son olarak gelelim hepinizin merak ettiği kısma! Bu kart ne kadar ediyor? Evet, her zaman olduğu gibi "İyi mal iyi para eder" kuralı burada da bozulmuyor ve...:

SID'siz fiyat : €169
1 SID çipiyle beraber : €189

Tabii eğer daha önceden bir HardSID ya da HardSID Quattro ISA kartı almışsanız, bu kartınızın arkasındaki seri numarasını bildirerek 10 Euro'luk bir indirim kazanabiliyorsunuz. Biz kartı SID'siz almanızı öneriyoruz. (Bu aynı zamanda firmanın da önerisi. Böylece kart ucuzluğunun yanında daha da hızlı geliyormuş.) Etrafınızdan kolayca (ve UCUZA!) bulabileceğiniz herhangi bir C64'ün içindeki SID çipini söküp kartınızla kullanabilirsiniz. SID çipleri soketli oldukları için takıp çıkarma sırasında C64'ünüze bir zarar vermeniz pek olası değildir (Tabii ki sadece yapmanız gerekeni yaptığınız sürece, sonra bizi sorumlu tutmayın!). C64'ü nereden alacağınızı sorarsanız, bunun için size en kolay yer olarak bit pazarını gösterebiliriz. Hades birkaç ay önce 250 bin liraya bir C64 almıştı!! Ayrıca zamanında Commodore alım/satımı yapmış olan bazı eski bilgisayarcılarda da C64 bulmanız olası. Olmazsa çevrenizdekileri soruşturun, mutlaka C64 sahibi biri çıkacaktır! Yine olmazsa Hades'in 4 C64'ünden birini isteyebilirsiniz!! Her neyse geyiği bir kenara bırakırsak, bu durumda sırf bir SID çipi için ayrıca 20 Euro fiyat farkı ödemek çok pahalıya geliyor (Bu arada firma kartlara en fazla bir tane SID çipi yerleştiriyor, geri kalanını kendiniz bulmalısınız).

- Kartla ilgili her türlü bilgi ve satın almak için http://www.hardsid.com/ adresine uğrayın.
- Kullanım kılavuzunu yine yukarıdaki adresten indirebilirsiniz. Kartla beraber herhangi bir kitapçık gelmiyor. http://hardsid.com/modules.php?name=Downloads&d_op=getit&lid=2
- http://www.hardsid.com/modules.php?name=Downloads&d_op=viewdownload&cid=7 adresinde kartla yapılmış bazı MIDI örnekleri mevcut.
- CBMZone sitesi (http://www.cbmzone.com/), HSQ PCI hakkında bir tanıtım yazısı yayınlamış. Kartı alıp kullanmak isteyenlere bu yazıyı okumalarını öneriyoruz: http://www.cbmzone.com/cgi-bin/index.pl?action=viewnews&id=5

ASSEMBLER KURSU-3

Assembler kursunun bu bölümünde önce 6510 komutlarını alfabetik olarak sıralayıp ne iş yaptıklarını öğreneceğiz. Daha sonra her komutu çeşitleriyle birlikte göreceğiz.

KOMUT LİSTESİ (ALFABETİK SIRA)

KOMUT	AÇIKLAMA
ADC	Akümdeki değerle sabit bir sayıyı veya adresteki değeri elde bitini de kullanarak topla.
AND	Akümdeki değerle, sabit bir sayı veya adresteki değer arasında "AND" işlemi yap.
ASL	Akümdeki değeri veya adresteki değeri bir bit sola kaydır.
BCC	Elde yoksa (C biti "0" ise) dallan.
BCS	Elde varsa (C biti "1" ise) dallan.
BEQ	Sonuç sıfır ise (Z biti "1" ise) dallan.
BIT	Adresteki bitleri Akü ile karşılaştır.
BMI	Sonuç eksi ise (N biti "1" ise) dallan.
BNE	Sonuç sıfır değilse (Z biti "0" ise) dallan.
BPL	Sonuç pozitif ise (N biti "0" ise) dallan.
BRK	Program içinde kesinti oluşturur. Kontrol başka bir programa geçer.
BVC	Taşma yoksa (V biti "0" ise) dallan.
BVS	Taşma varsa (V biti "1" ise) dallan.
CLC	Elde bitini "0" yap.
CLD	Ondalık moddan çık. (D bitini "0" yap)
CLI	Kesintileri serbest bırak. (I bitini "0" yap)
CLV	Taşma bitini "0" yap.
CMP	Akümdeki değeri sabit bir sayıyla veya adresteki değerle karşılaştır.
CPX	X registerindeki değeri sabit bir sayıyla veya adresteki değerle karşılaştır.
CPY	Y registerindeki değeri sabit bir sayıyla veya adresteki değerle karşılaştır.
DEC	Adresteki değeri 1 azalt.
DEX	X registerindeki değeri 1 azalt.
DEY	Y registerindeki değeri 1 azalt.
EOR	Akümdeki değerle, sabit bir sayı veya adresteki değer arasında "EOR" işlemi yap.
INC	Adresteki değeri 1 arttır.
INX	X registerindeki değeri 1 arttır.
INY	Y registerindeki değeri 1 arttır.
JMP	Program içindeki vaya hafızadaki başka bir adrese git.
JSR	Geri dönüş adresini saklayarak program içindeki veya hafızadaki başka bir adrese git.
LDA	Aküye sabit bir sayıyı veya adresteki değeri yükle.
LDX	X registerine sabit sayıyı veya adresteki değeri yükle.
LDY	Y registerine sabit sayıyı veya adresteki değeri yükle.
LSR	Akümdeki veya adresteki değeri bir bit sağa kaydır.
NOP	Hiç bir işlem yapma.
ORA	Akümdeki değerle, sabit bir sayı veya adresteki değer arasında "OR" işlemi yap.
PHA	Akümdeki değeri yığına at.

PHP	6510'un statü registerinin değerini yığına at.
PLA	Yığındaki değeri Aküye geri yükle.
PLP	Yığındaki değeri statü registerine geri yükle.
ROL	Aküdeki veya adresteki değeri bir bit sola döndür.
ROR	Aküdeki veya adresteki değeri bir bit sağa döndür.
RTI	Kesinti (İnterrupt) işleminden geri dön.
RTS	Alt programdan geri dön.
SBC	Aküdeki değerden sabit bir sayıyı veya adresteki değeri elde bitini de kullanarak çıkar.
SEC	Elde bitini "1" yap.
SED	Ondalık moda gir. (D bitini "1" yap.)
SEI	Kesintileri durdur. (I bitini "1" yap.)
STA	Aküdeki değeri adrese yerleştir.
STX	X registerindeki değeri adrese yerleştir.
STY	Y registerindeki değeri adrese yerleştir.
TAX	Aküdeki değeri X registerine kopyala.
TAY	Aküdeki değeri Y registerine kopyala.
TSX	Yığın Göstergecinin (SP) değerini X registerine kopyala.
TXA	X registerinin değerini Aküye kopyala.
TXS	X registerinin değerini Yığın Göstergecine (SP) kopyala.
TYA	Y registerinin değerini Aküye kopyala.

"AND" "OR" VE "EOR" İŞLEMLERİ

Bu dersimizde önce "AND" "OR" "EOR" ve aritmetiksel işlemlerinin mantığını öğreneceğiz. Daha sonra ise komutları ayrıntılı bir şekilde göreceğiz.

"AND", "OR" ve "EOR" işlemleri bit düzeyinde işlem yapar ve sadece Akü ile kullanılır. Bu komutlarla

herhangi bir adresteki değerin istediğiniz bitlerini "0" veya "1" yapabilirsiniz.

"AND" işlemi bitler arasında çarpma işlemi yapar. Eğer karşılaştırılan bitlerin ikisi de "1" ise sonuç "1" dir. Diğer durumlarda sonuç "0" dır. Bunu bir tablo ile görelim. AND işlemini herhangi bir biti "0" yapmak için kullanabilirsiniz.

BİTLER	İŞLEM	SONUÇ
0 - 0	**AND**	**0**
0 - 1	**AND**	**0**
1 - 0	**AND**	**0**
1 - 1	**AND**	**1**

"OR" işlemi bitler arasında toplama işlemi yapar. Eğer karşılaştırılan bitlerin ikiside "0" ise sonuç "0" dır. Diğer durumlarda sonuç "1" dir. Bunu bir tablo ile görelim.OR işlemini herhangi bir biti "1" yapmak için kullanabilirsiniz.

BİTLER	İŞLEM	SONUÇ
0 - 0	**OR**	**0**
0 - 1	**OR**	**1**
1 - 0	**OR**	**1**
1 - 1	**OR**	**1**

"EOR" işlemi diğerlerinden çok farklıdır. Eğer karşılaştırılan bitlerin ikiside aynı ise sonuç "0" dir. Herhangi biri diğerinden farklıysa sonuç "1" dir.Bunu bir tablo ile görelim.

BİTLER	İŞLEM	SONUÇ
0 - 0	EOR	0
0 - 1	EOR	1
1 - 0	EOR	1
1 - 1	EOR	0

Eğer EOR komutunu kullanmayı bilirseniz herhangi bir adresin değeri her seferinde başlangıç ve bitiş değeriyle değiştirebilirsiniz. Bir örnek verelim.

Diyelim ki bir program yaptınız ve program içinde bir arttırma komutunuz var. Bir sayaç ile artış miktarını kontrol ediyorsunuz. Sayaç belli bir sayıya ulaştığında arttırma komutunu azaltma komutuna çevirmeniz gerekiyor. Bunun için EOR komutunu kullanabilirsiniz. Her EOR komutu işlendiğinde komut kodu AZALTMA-ARTTIRMA şeklinde değişir. Aşağıdaki tabloyu inceleyin.

ETİKET	KOMUT	PARAMETRE	AÇIKLAMA
	...	.....	.....
DENEME	INC	$D000	Adresi 1 arttır
	LDX	SAYAÇ	Sayacı oku
	INX		Sayaç değerini 1 arttır
	CPX	#$40	Sayaç #$40 mı ?
	BNE	GİT	Değilse git.
	LDA	DENEME	Adresi oku. (Okunan ilk değer #$EE' dir.) (Son değer #$CE' dir.)
	EOR	#$20	#$20 ile EOR'la. (Sonuç #$CE olur.) (Daha sonra #$EE olur.)
	STA	DENEME	Adrese yaz.
	LDX	#$00	Sayacın yeni değeri
GİT	STX	SAYAÇ	Sayaca yaz
	...	.....	.....

TOPLAMA VE ÇIKARMA İŞLEMLERİ

Eğer bir program yazarken toplama veya çıkarma işlemi yapacaksanız dikkat etmeniz gereken bir nokta vardır. Bir toplama işlemine başlamadan önce mutlaka **CLC** komutu kullanarak ELDE bitini sıfırlayın. Daha sonra toplama işleminizi yapın. Yaptığınız toplama (eğer bir baytlıksa) işleminde sonuç 255' i geçerse (en fazla 510 olur) CARRY (Elde) biti "1" olur. Bu "1" ilk dersimizde gördüğünüz LOW BAYT - HIGH BAYT formatındaki HIGH BAYT'ı gösterir ve 256 demektir. Toplama işleminizin sonucu CARRY * 256 + AKÜDEKİ DEĞER şeklindedir. Aküdeki değer LOW BAYT anlamına gelir.

Bu anlattıklarımızı bir örnekle görelim.

ETİKET	KOMUT	PARAMETRE	AÇIKLAMA
	...	.....	.....
	LDX	#$20	X'e #$20 değerini yükle.
	STX	ADRES1	Bu değeri adres1'e kopyala.
	LDA	#$80	Aküye #$80 değerini yükle.
	CLC		Elde bitini "0" yap.
	ADC	ADRES1	Aküdeki değeri Adres1 deki değer ile topla.
	STA	ADRES2	Sonucu Adres2' ye yaz.
	BCS	GİT	Eğer ELDE biti "1" ise git.
	RTS		ELDE yoksa programı bitir.
GİT	...	.....	.....

Eğer bir çıkarma işlemi yapacaksanız bu kezde **SEC** komutu kullanarak ELDE bitini "1" yapın. Daha sonra çıkarma işleminizi yapın. Bunun faydası AKÜ - SAYI işleminde SAYI Aküdeki değerden büyükse sonucun doğru olmasını sağlar. Şunu unutmayın: Çıkarma işleminde dikkatli olun.

KOMUT LİSTESİ (ALFABETİK SIRA)

6510 işlemcisine ait komutları önce ikiye ayırmak gerekiyor.

1- Standart Komutlar : Bu kategorideki komutlar 6510' un üreticisi tarafından çalışması garanti edilen komutlardır.

2 - Standart Olmayan Komutlar : Bu kategorideki komutlar için her makinada çalışabilir diye kural yoktur. Sizin makinanızda çalışan bir komut başka bir makinada kilitlenmeye yol açabilir. Ama bu kategorideki komutlardan bazıları çok işe yarayabilir. Yazı dizimizin sonunda bu komutların bir listesini vermeyi düşünüyorum

NOT : Komut Tablolarında **$ADR** Sıfırıncı sayfa'daki bir adresi gösterir 0 ile 255 ($00 - $FF) arasındaki adresleri kapsar. (0 ve 255 dahil) **$ADRES** ise 0 ile 65535 ($0000 - $FFFF) arasındaki adresleri kapsar. (0 ve 65535 dahil) **#$SAYI** 0 ile 255 ($00 - $FF) arasındaki bir sayıyı gösterir. (0 ve 255 dahil)

KOMUT	AÇIKLAMA			ETKİLENEN BAYRAKLAR
ADC	Aküdeki değerle sabit bir sayıyı veya başka bir adresteki değeri elde bitini de kullanarak topla. Sonucu aküye yaz.			N Z C V
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	ADC #$SAYI	69	2	2
Sıfırıncı Sayfa	ADC $ADR	65	2	3
Sıfırıncı Sayfa X indeksli	ADC $ADR, X	75	2	4
Mutlak	ADC $ADRES	6D	3	4
Mutlak X İndeksli	ADC $ADRES, X	7D	3	4 *
Mutlak Y İndeksli	ADC $ADRES, Y	79	3	4 *
İndeksli Dolaylı	ADC ($ADR, X)	61	2	6
Dolaylı İndeksli	ADC ($ADR), Y	71	2	5 *

* Sayfa sınırı ile karşılaştığında 1 ekle

KOMUT	AÇIKLAMA			ETKİLENEN BAYRAKLAR
AND	Aküdeki değerle sabit bir sayı veya adresteki değer arasında "AND" işlemi yap. Sonucu aküye yaz.			N Z
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	AND #$SAYI	29	2	2
Sıfırıncı Sayfa	AND $ADR	25	2	3
Sıfırıncı Sayfa X indeksli	AND $ADR, X	35	2	4
Mutlak	AND $ADRES	2D	3	4
Mutlak X İndeksli	AND $ADRES, X	3D	3	4 *
Mutlak Y İndeksli	AND $ADRES, Y	39	3	4 *
İndeksli Dolaylı	AND ($ADR, X)	21	2	6
Dolaylı İndeksli	AND ($ADR), Y	31	2	5

* Sayfa sınırı ile karşılaştığında 1 ekle

KOMUT	AÇIKLAMA			ETKİLENEN BAYRAKLAR
ASL	Aküdeki veya adresteki değeri bir bit sola kaydır (Sayı 2 ile Çarpılmış olur). Sayının 7. biti "C" 'ye yüklenir, 0. Bitine herzaman "0" değeri yüklenir.			N Z C
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Akü	ASL A	0A	1	2
Sıfırıncı Sayfa	ASL $ADR	06	2	5
Sıfırıncı Sayfa X indeksli	ASL $ADR, X	16	2	6
Mutlak	ASL $ADRES	0E	3	6
Mutlak X İndeksli	ASL $ADRES, X	1E	3	7

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
BCC	Elde biti "0" ise dallan. (C = 0 ise dallan)		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Relative	BCC #$SAYI	90	2	2 **

** Dallanma aynı sayfada ise 1 ekle. Eğer dallanma başka bir sayfaya ise 2 ekle.

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
BCS	Elde biti "1" ise dallan. (C = 1 ise dallan)		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Relative	BCS #$SAYI	B0	2	2 **

** Dallanma aynı sayfada ise 1 ekle. Eğer dallanma başka bir sayfaya ise 2 ekle.

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
BEQ	Sonuç sıfır ise dallan. (Z = 1 ise dallan)		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Relative	BEQ #$SAYI	F0	2	2 **

** Dallanma aynı sayfada ise 1 ekle. Eğer dallanma başka bir sayfaya ise 2 ekle.

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
BIT	Bellekteki Bitleri Akü İle Karşılaştır. (Akü ile Bellek Arasında AND işlemi yap. Bellekteki sayının 7. ve 6. bitlerini STATUS registerinin 7. ve 6. bitlerine yerleştirilir. Eğer A AND BELLEK işleminin sonucu 0 ise Z = 1 olur, değilse Z = 0 olur.		Z 7.bit - --> N'ye 6.bit ---> V'ye	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Sıfırıncı Sayfa	BIT $ADR	24	2	3
Mutlak	BIT $ADRES	2C	3	4

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
BMI	Sonuç eksi ise dallan. (N = 0 ise dallan)		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Relative	BMI #$SAYI	30	2	2 **

** Dallanma aynı sayfada ise 1 ekle. Eğer dallanma başka bir sayfaya ise 2 ekle.

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
BNE	Sonuç eksi değilse dallan. (Z = 0 ise dallan)		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Relative	BNE #$SAYI	D0	2	2 **

** Dallanma aynı sayfada ise 1 ekle. Eğer dallanma başka bir sayfaya ise 2 ekle.

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
BPL	Sonuç Pozitif ise dallan. (N = 0 ise dallan)		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Relative	BPL #$SAYI	10	2	7 **

** Dallanma aynı sayfada ise 1 ekle. Eğer dallanma başka bir sayfaya ise 2 ekle.

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
BRK	Kesinti meydana getirir. Program sayacı ve İşlemci Statü registeri yığına atılır. Mikroişlemci $0316 vektör adresi üzerinden $FE66 adresindeki programı çalıştırır. "BRK komutu 'I' biti "1" yapılarak önlenemez"		I	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	BRK	00	1	7

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
BVC	Taşma biti "0" ise dallan. (V = 0 ise dallan)		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Relative	BVC #$SAYI	50	2	2 **

** Dallanma aynı sayfada ise 1 ekle. Eğer dallanma başka bir sayfaya ise 2 ekle.

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
BVS	Taşma biti "1" ise dallan. (V = 0 ise dallan)		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Relative	BVS #$SAYI	70	2	2 **

** Dallanma aynı sayfada ise 1 ekle. Eğer dallanma başka bir sayfaya ise 2 ekle.

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
CLC	Elde bitini "0" yap. (C = 0 yap)		C	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	CLC	18	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
CLD	Ondalık moddan çık. (D = 0 yap.)		D	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	CLD	D8	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
CLI	Kesinti (İnterrupt) önleme bitini Sıfırla (İnterruptları serbest bırak) (I = 0 yap)		I	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	CLI	58	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
CLV	Taşma bitini sıfırla (V = 0 yap)		V	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	CLV	B8	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
CMP	Akümdeki değerle sabit bir sayıyı veya adresteki değeri karşılaştır.		N Z C	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	CMP #$SAYI	C9	2	2
Sıfırıncı Sayfa	CMP $ADR	C5	2	3
Sıfırıncı Sayfa X indeksli	CMP $ADR, X	D5	2	4
Mutlak	CMP $ADRES	CD	3	4
Mutlak X İndeksli	CMP $ADRES, X	DD	3	4 *
Mutlak Y İndeksli	CMP $ADRES, Y	D9	3	4 *
İndeksli Dolaylı	CMP ($ADR, X)	C1	2	6
Dolaylı İndeksli	CMP ($ADR), Y	D1	2	5 *

Sayfa sınırı ile karşılaştığında 1 ekle

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
CPX	X registerindeki değerle sabit bir sayıyı veya adresteki değeri karşılaştır.		N Z C	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	CPX #$SAYI	E0	2	2
Sıfırıncı Sayfa	CPX $ADR	E4	2	3
Mutlak	CPX $ADRES	EC	3	4

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
CPY	Y registerindeki değerle sabit bir sayıyı veya adresteki değeri karşılaştır.		N Z C	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	CPY #$SAYI	C0	2	2
Sıfırıncı Sayfa	CPY $ADR	C4	2	3
Mutlak	CPY $ADRES	CC	3	4

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
DEC	Bellekteki sayıyı 1 azalt. Sayı 0'sa DEC komutundan sonra Adresteki değer 255 ($FF) olur.		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Sıfırıncı Sayfa	DEC $ADR	C6	2	5
Sıfırıncı Sayfa X indeksli	DEC $ADR, X	D6	2	6
Mutlak	DEC $ADRES	CE	3	6
Mutlak X İndeksli	DEC $ADRES, X	DE	3	7

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
DEX	X registerindeki değeri 1 azalt. X = 0 iken DEX komutundan sonra X' in yeni değeri 255 ($FF) olur.		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	DEX	CA	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
DEY	Y registerindeki değeri 1 azalt. Y = 0 iken DEY komutundan sonra Y' nin yeni değeri 255 ($FF) olur.		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	DEY	88	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
EOR	Akü ile sabit bir sayı veya adresteki değer arasında "Exlusıve-OR" işlemi yap. Sonucu Aküye yaz.		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	EOR #$SAYI	49	2	2
Sıfırıncı Sayfa	EOR $ADR	45	2	3
Sıfırıncı Sayfa X indeksli	EOR $ADR, X	55	2	4
Mutlak	EOR $ADRES	4D	3	4
Mutlak X İndeksli	EOR $ADRES, X	5D	3	4 *
Mutlak Y İndeksli	EOR $ADRES, Y	59	3	4 *
İndeksli Dolaylı	EOR ($ADR, X)	41	2	6
Dolaylı İndeksli	EOR ($ADR), Y	51	2	5 *

Sayfa sınırı ile karşılaştığında 1 ekle

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
INC	Bellekteki sayıyı 1 arttır.. Sayı 255 ($FF) ise DEC komutundan sonra Adresteki değer 0 olur.		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Sıfırıncı Sayfa	INC $ADR	E6	2	5
Sıfırıncı Sayfa X indeksli	INC $ADR, X	F6	2	6
Mutlak	INC $ADRES	EE	3	6
Mutlak X İndeksli	INC $ADRES, X	FE	3	7

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
INX	X registerindeki değeri 1 arttır. X = $FF iken INX komutundan sonra X' in yeni değeri 0 olur.		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	INX	E8	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
INY	Y registerindeki değeri 1 arttır. Y = $FF iken INY komutundan sonra Y' nin yeni değeri 0 olur.		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	INY	C8	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
JMP	Programın çalışması verilen adresten devam eder.		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Mutlak	JMP $ADRES	4C	3	3
Dolaylı	JMP ($ADRES)	6C	3	5

Dolaylı JMP komutunda programın çalışmaya devam edeceği adres LO-BAYT, HI-BAYT şeklinde ADRES' te ve ADRES + 1'de tutulur.

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
JSR	Program o anki çalışma adresini yığına atarak başka bir adresten çalışmaya devam eder. Gidilen adreste RTS komutuna rastlayan program, daha önce sakladığı adresi yığından geri alarak çalışmasına, JSR komutundan bir sonraki komuttan itibaren çalışmasına devam eder. Her JSR komutuna karşılık mutlaka 1 adet RTS komutu bulunmalıdır.		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Mutlak	JSR $ADRES	20	3	6

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
LDA	Aküye sabit bir sayı veya adresteki değer yüklenir.		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	LDA #$SAYI	A9	2	2
Sıfırıncı Sayfa	LDA $ADR	A5	2	3
Sıfırıncı Sayfa X indeksli	LDA $ADR, X	B5	2	4
Mutlak	LDA $ADRES	AD	3	4
Mutlak X İndeksli	LDA $ADRES, X	BD	3	4 *
Mutlak Y İndeksli	LDA $ADRES, Y	B9	3	4 *
İndeksli Dolaylı	LDA ($ADR, X)	A1	2	6
Dolaylı İndeksli	LDA ($ADR), Y	B1	2	5 *

* Sayfa sınırı ile karşılaştığında 1 ekle

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
LDX	X registerine sabit bir sayı veya adresteki değer yüklenir.		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	LDX #$SAYI	A2	2	2
Sıfırıncı Sayfa	LDX $ADR	A6	2	3
Sıfırıncı Sayfa Y indeksli	LDX $ADR, Y	B6	2	4
Mutlak	LDX $ADRES	AE	3	4
Mutlak Y İndeksli	LDX $ADRES, Y	BE	3	4 *

* Sayfa sınırı ile karşılaştığında 1 ekle

KOMUT	AÇIKLAMA			ETKİLENEN BAYRAKLAR
LDY	Y registerine sabit bir sayı veya adresteki değer yüklenir.			N Z
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	LDY #$SAYI	A0	2	2
Sıfırıncı Sayfa	LDY $ADR	A4	2	3
Sıfırıncı Sayfa X indeksli	LDY $ADR, X	B4	2	4
Mutlak	LDY $ADRES	AC	3	4
Mutlak X İndeksli	LDY $ADRES, X	BC	3	4 *

* Sayfa sınırı ile karşılaştığında 1 ekle

KOMUT	AÇIKLAMA			ETKİLENEN BAYRAKLAR
LSR	Aküdeki veya adresteki değeri bir bit sağa kaydır (Sayı 2'ye bölünmüş olur) . Sayının 0. biti "C" 'ye yüklenir, 7. bitine herzaman "0" değeri yüklenir.			N = 0 Z C
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Akü	LSR A	4A	1	2
Sıfırıncı Sayfa	LSR $ADR	46	2	5
Sıfırıncı Sayfa X indeksli	LSR $ADR, X	56	2	6
Mutlak	LSR $ADRES	4E	3	6
Mutlak X İndeksli	LSR $ADRES, X	5E	3	7

KOMUT	AÇIKLAMA			ETKİLENEN BAYRAKLAR
NOP	Hiç bir işlem yapma			----------
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	NOP	EA	1	2

KOMUT	AÇIKLAMA			ETKİLENEN BAYRAKLAR
ORA	Aküdeki değerle sabit bir sayı veya adresteki değer arasında "OR" işlemi yap. Sonucu aküye yaz.			N Z
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	ORA #$SAYI	09	2	2
Sıfırıncı Sayfa	ORA $ADR	05	2	3
Sıfırıncı Sayfa X indeksli	ORA $ADR, X	15	2	4
Mutlak	ORA $ADRES	0D	3	4
Mutlak X İndeksli	ORA $ADRES, X	1D	3	4 *
Mutlak Y İndeksli	ORA $ADRES, Y	19	3	4 *
İndeksli Dolaylı	ORA ($ADR, X)	01	2	6
Dolaylı İndeksli	ORA ($ADR), Y	11	2	7

* Sayfa sınırı ile karşılaştığında 1 ekle

KOMUT	AÇIKLAMA			ETKİLENEN BAYRAKLAR
PHA	Aküyü yığına at.			----------
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	PHA	48	1	3

KOMUT	AÇIKLAMA			ETKİLENEN BAYRAKLAR
PHP	İşlemcinin statü registerini yığına at.			----------
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	PHP	08	1	3

KOMUT	AÇIKLAMA			ETKİLENEN BAYRAKLAR
PLA	Aküyü yığından geri al.			N Z
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	PLA	68	1	4

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
PLP	İşlemcinin Statü registerini yığından geri al.		YIĞINDAN	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	PLP	28	1	4

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
ROL	Aküdeki veya adresteki değeri bir bit sola döndür. "C" biti sayının 0. bit'ine yüklenirken, sayının 7. bit'i "C" bitine yüklenir.		N Z C	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Akü	ROL A	2A	1	2
Sıfırıncı Sayfa	ROL $ADR	26	2	5
Sıfırıncı Sayfa X indeksli	ROL $ADR, X	36	2	6
Mutlak	ROL $ADRES	2E	3	6
Mutlak X İndeksli	ROL $ADRES, X	3E	3	7

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
ROR	Aküdeki veya adresteki değeri bir bit sağa döndür. "C" biti sayının 7. bit'ine yüklenirken, sayının 0. bit'i "C" bitine yüklenir.		N Z C	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Akü	ROR A	6A	1	2
Sıfırıncı Sayfa	ROR $ADR	66	2	5
Sıfırıncı Sayfa X indeksli	ROR $ADR, X	76	2	6
Mutlak	ROR $ADRES	6E	3	6
Mutlak X İndeksli	ROR $ADRES, X	7E	3	7

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
RTI	İnterrupt (Kesinti) işleminden geri dön.		YIĞINDAN	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	RTI	40	1	6

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
RTS	Alt programdan geri dön. Veya ana programı bitir		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	RTS	60	1	6

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
SBC	Aküden sabit bir sayıyı veya adresteki değeri "C" (Elde) bitini de kullanarak çıkart. Sonucu Aküye yaz.		N Z C V	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	SBC #$SAYI	E9	2	2
Sıfırıncı Sayfa	SBC $ADR	E5	2	3
Sıfırıncı Sayfa X indeksli	SBC $ADR, X	F5	2	4
Mutlak	SBC $ADRES	ED	3	4
Mutlak X İndeksli	SBC $ADRES, X	FD	3	4 *
Mutlak Y İndeksli	SBC $ADRES, Y	F9	3	4 *
İndeksli Dolaylı	SBC ($ADR, X)	E1	2	6
Dolaylı İndeksli	SBC ($ADR), Y	F1	2	5 *

* Sayfa sınırı ile karşılaştığında 1 ekle

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
SEC	Elde bitini "1" yap. (C = 1 yap)		C	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	SEC	38	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
SED	Ondalık moda gir. (D = 1 yap)		D	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	SED	F8	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
SEI	İnterruptları durdur. (I = 1 yap)		I	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	SEI	78	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
STA	Akümdeki değeri adrese yaz.		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Sıfırıncı Sayfa	STA $ADR	85	2	2
Sıfırıncı Sayfa X indeksli	STA $ADR, X	95	2	4
Mutlak	STA $ADRES	8D	3	4
Mutlak X İndeksli	STA $ADRES, X	9D	3	5
Mutlak Y İndeksli	STA $ADRES, Y	99	3	5
İndeksli Dolaylı	STA ($ADR, X)	81	2	6
Dolaylı İndeksli	STA ($ADR), Y	91	2	6

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
STX	X registerindeki değeri adrese yaz.		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Sıfırıncı Sayfa	STX $ADR	86	2	3
Sıfırıncı Sayfa Y indeksli	STX $ADR, Y	96	2	4
Mutlak	STX $ADRES	8E	3	4

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
STY	Y registerindeki değeri adrese yaz.		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Sıfırıncı Sayfa	STY $ADR	84	2	3
Sıfırıncı Sayfa X indeksli	STY $ADR, X	94	2	4
Mutlak	STY $ADRES	8C	3	4

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
TAX	Akümdeki değeri X registerine kopyala.(X = A olur.)		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	TAX	AA	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
TAY	Akümdeki değeri Y registerine kopyala.(Y = A olur.)		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	TAY	A8	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
TSX	Yığın göstergecini X registerine kopyala. (X = SP olur.)		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	TSX	BA	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
TXA	X registerini Aküye kopyala. (A = X olur.)		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	TXA	8A	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
TXS	X registerini Yığın Göstergecine kopyala. (SP = X olur.) Tehlikeli bir komuttur.		----------	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	TXS	9A	1	2

KOMUT	AÇIKLAMA		ETKİLENEN BAYRAKLAR	
TYA	Y registerini Aküye kopyala. (A = Y olur.)		N Z	
ADRESLEME MODU	**ASSEMBLER KOMUTU**	**İŞLEM KODU**	**BAYT SAYISI**	**ÇEVRİM SAYISI**
Dolaysız	TYA	98	1	2

Assembler bölümünü böylece bitirmiş olduk. Bir program yazmaya başlamadan önce programda ne gibi özellikler bulunacak, nasıl çalışacak vs. bunları belirleyin. Programı yazarken hiç acele etmeyin. İlk önce programın ekran düzenini kafanızda canlandırın. Ekranda yazı nerede bulunacak, renkler ne olacak bunları ayarlayın. Daha sonra programı yazmaya başlayın, yazdıkça arada bir kaydedin. Daha sonra çalıştırın ama şuna dikkat edin. Henüz bitmemiş bir programınız olduğu için programı çalıştırınca bilgisayarınız kilitlenebilir veya istediğiniz gibi çalışmayabilir. Bir örnek verelim isterseniz.
Diyelim ki programda müzik çalacak ve bunun için LDA #$00 JSR $1000 gibi bir komut vererek hazırlık yaptınız. (İntrocular iyi bilir bu komutları ☺) Fakat hafızaya müzik yüklemeyi unuttuysanız C64 kilitlenecektir. Veya değişik bir font kullanacaksınız ve font hafızaya yüklenmiş durumda. Programda yeni karakter setinin bulunduğu bölgeyi VIC'e bildirdiniz fakat ekran birden saçma sapan karakterlerle doldu. Sebebi ise $D018 adresine yanlış bir sayı yerleştirilmiş olabilir (belki de karakter setiniz bozulmuştur). İşte bunun gibi durumlarla karşılaşmamak için eğer ekstradan font, müzik, sprite vs. kullanacaksanız önceden bunları hafızaya yükleyin sonradan programda aktif hale getirin. Size tavsiyem bunların hepsini tek seferde kaydedin. Yani fontu ayrı müziği ayrı kaydetmeyin. Büyük bir ihtimalle – neredeyse standart diyebilirim- müzik $1000-$2000 ve font ise (1x1 font için) $2000-$2200 adresleri arasında olacaktır. Code için ise $0900-$1000 arasını ve daha üst adresleri kullanabilirsiniz. Eğer programınızda logo falan olacaksa size tavsiyem $4000'den yukarısını kullanın demek olacaktır.

Bazen basit bir hata programınızda büyük bir problem ortaya çıkarabilir. Mesela gereksiz yere çağrılan bir alt rutin, başka bir alt rutin tarafından kullanılan hafıza bölgesini değiştiriyor olabilir, siz de niye çalışmıyor diye kafayı sıyırırsınız. O nedenle assembler ile program yazarken dikkat etmelisiniz. Assembler ile yazılmış bir programda hata bulmak zordur. Ama programları yazdıkça, zamanla elinizde neredeyse klişeleşmiş rutinler olacaktır ve işiniz daha da kolaylaşacaktır.

Son olarak şunu söyleyeyim. Eğer başkasının yazdığı programları incelerseniz çok faydasını görebilirsiniz. En azından bazı işlerin nasıl yapıldığını anlayabilirsiniz ama şunu açıkça söylemekte fayda var. Bir başkasının yazdığı assembler programları anlamak veya çözmek her zaman kolay değildir. Mesela incelediğiniz programda sürekli olarak JSR komutlarıyla alt rutinlere sıçrama yapılıyorsa bir süre sonra hangi adresin nereden çağrıldığını karıştırırsınız. Veya programcı ROM'lardaki rutinleri sık sık kullanıyor olabilir. Bunun için de C64'ün ROM'larını incelemeniz ve oradaki rutinlerin ne iş yaptığını bilmeniz gereklidir.

Bu kadar laftan sonra artık bişeyler yaparsınız değil mi?
Hadi bakalım kolay gelsin.

C64 ASM V1.1

Assemblerin biri bitiyor diğeri başlıyor değil mi? 12 sayfalık bir bölümden sonra bilmem kaç sayfalık C64ASM bölümüne başlıyoruz. Korkmanıza gerek yok, sayı sistemlerini komutları vs. yeniden anlatmayacağım. C64ASM PC'de 6510 assembler komutlarını kullanarak program yazmamıza yarayan yardımcı programlardan biri. Eğer PC'nizde VICE gibi bir emülatör programınız varsa C64ASM ile elde ettiğiniz **.PRG** uzantılı dosyayı hemen çalıştırabilirsiniz. C64ASM programı gerçekten çok kullanışlı bir programdır. Ben artık bu programı kullanıyorum. En güzel tarafı C64'ün hafızası tamamen kullanıma açık olmasıdır. Yani emülatör ile mesela TURBOASSEMBLER kullanınca karşınıza çıkan hafıza sınırlaması –TASM ile $C000'dan başlayan bir program yazamazsınız- C64ASM'de yoktur. Bu programın gerçekten güzel özellikleri vardır ama her güzelin bir kusuru bulunur dedikleri gibi bu programın kusuru da **MS-DOS KOMUT İSTEMİ'**nden çalıştırılıyor olmasıdır. İnternette o kadar aradığım halde Windows altında çalışan bir C64ASSEMBLER EDİTOR programı bulamadım. Bir program yazmak için size gerekli olan WORDPAD programıdır. WORDPAD ile yazdığınız programı **.asm** uzantılı olarak mutlaka **C64ASM** programının olduğu klasöre kaydedin. Bu klasörün masaüstünde olmasını tavsiye ederim. Sonra ise **BAŞLAT → PROGRAMLAR → MSDOS KOMUT İSTEMİ'**ni çalıştırın. Daha sonra komut satırından **CD DESKTOP → CD klasörünadı →C64ASM programınadı** yazarak programınız derlenir ve size default olarak **programadı.PRG** olarak bir c64 file üretilir. DİKKAT : programın adı en fazla 8 karakter olabilir. (MS DOS zamanlarından kalma bir sınrlama ☹☹)

Programın 1.1 versiyonunu www.students.itu.edu.tr/~celikdeni/c64asm.zip adresinde bulabilirsiniz. Programın yazarı **Balint Toth** isimli bir Macar'dır. Lafı fazla uzatmadan C64ASM1.1 klasöründe bulunan **C64ASM.EXE** programının özelliklerini ve kullanılışını anlatmaya başlayabiliriz. Program SHAREWARE'dir.

C64ASM'ün kendisi DOS'ta çalışır ve COMMODORE 64'ün makine dilinde dosyalar oluşturur. C64ASM, program içinde kullanılan etiketleri, sembolik sabitleri, matematiksel ifadeleri, dosya eklemeyi destekler ve gerekirse ayrıntılı rapor yaratabilir. PRG, T64 ve P00 uzantılı dosya tiplerini C64ASM ile kullanbilirsiniz. Komutun kullanım şekli aşağıdaki gibidir. Köşeli parantezlerin içindekilerinin kullanılması zorunlu değildir.

C64ASM kaynakdosyaadı[.asm] [çıkışdosyaadı[.prg]] [opsiyonlar]

Kaynakdosyaadı : WORD ile yazmış olduğunuz ve .ASM uzantılı olarak kaydettiğiniz dosyanın adı.
Çıkışdosyaadı : C64ASM ile oluşan dosyanın adını ve dosya tipini burada tanımlayabilirsiniz. Default olarak tanımlanan çıkış dosya tipi .PRG'dir ve bunu .T64 veya .P00 olarak değiştirebilirsiniz.
Opsiyonlar :
/S : Rapor dosyasına sembol tablosu yazılır.
/L : Rapor dosyasına derleme tablosu yazılır.
/T : Rapor dosyasına özet yazılır.
/R = Rapordosyasıadı [.rep] : Rapor dosyası adı için başka bir isim verilebilir.
/6 : Çıkış dosyası tipi olarak her zaman T64 kullanmak için kullanılır.
/0 : Çıkış dosyası tipi olarak her zaman P00 kullanmak için kullanılır.

ÖRNEK : C64ASM main /SLT komutuyla main.asm isimli programınız main.prg olarak derlenir ve maksimum ayrıntılı olarak main.rep dosyası oluşturulur. Eğer ayrıntılarla ilgilenmiyorsanız sadece **C64ASM programadı** komutu işinizi görür.

Program 6510'un standart komutlarını ve adresleme modlarını desteklemektedir. Derleyici küçük / büyük harf ayrımına duyarsızdır. Yani programda küçük ve büyük harfleri karışık kullanabilirsiniz. Program yazarken her satırda mutlaka komut olacak diye bir kural yoktur. " **;** " (noktalı virgül) ile başlayan satırlar ve açıklamalar derleyici tarafından dikkate alınmaz.

Temel yazım şekli aşağıdaki gibidir. Köşeli parantezlerin içindekilerinin kullanılması zorunlu değildir.

[ETİKET] KOMUT [PARAMETRE] [; AÇIKLAMA]

Parametrenin olup olmaması komuta bağlıdır.
Şimdi bunları sırayla açıklayalım.

ETİKET : Etiket bir bellek adresinin sembolik adıdır. Ayrıca global ve local değişkenler de etiket kullanarak tanımlanabilir. Eğer isterseniz "_" (altçizgi) karakterini de kullanabilirsiniz.

ÖRNEKLER

```
          -1-                  -2-                  -3-                  -4-

          .....                .....                .....                lda #<_yazı
tekrar    inc $fb              inc renk             lda sayı             ldy #>_yazı
          bne tekrar           .....                .....                jsr print
          .....
                     renk  =$d020       sayı  .byte $30     _yazı  .text "c64 türkiye"
                                                            print  =$ab1e
```

1. örnekte "**tekrar"** etiketi programda döngü amacıyla kullanılmıştır. Derleme sonunda bulunan değer BNE komutunun parametresi olur.
2. örnekte ise **"renk"** etiketi $d020 adresinin yerine kullanılmıştır ve global bir değişken olarak tanımlanmıştır. Derleme sırasında program içinde yer alan ve "**renk"** ifadesinin kullanıldığı her komuttan sonra $20 ve $d0 değerleri yerleştirilir.
3. örnekte **"sayı"** etiketi programda bulunduğu adreste default olarak $30 değerine sahip bir değişken olarak tanımlanmıştır.
4. örnekte ise altçizgili bir kullanıma örnek olarak verilmiştir. "c64 türkiye" yazısının hafızadaki adresini göstermektedir.

Etiket kullanmanın faydası şudur. Program içinde bir kez tanımladıktan sonra, programda herhangi bir ekleme veya çıkarma yapsanız bile programın çalışması değişmeyecektir. Ayrıca isimleri hatırlamak sayıları hatırlamaktan daha kolaydır.

KOMUT : Yazdığınız programda kullanacağınız makine dili komutları **"LDA"**, **"NOP"** gibi 6510 komutları veya C64ASM'ye ait olan ve **"." (nokta)** ile başlayan özel komutlar olabilir.

PARAMETRE : Parametre olarak komuta ait herhangi bir adresleme modu olabilir. LDA adres,X gibi. Veya daha sonra bahsedeceğimiz çeşitli aritmetiksel ifadeler olabilir. Sayı kullanılacaksa bu sayı Binary, Hexadecimal veya Decimal olabilir.

AÇIKLAMA: Açıklama olarak programın o kısmında ne yaptığınızı veya o komutu niçin kullandığınızı yazabilirsiniz. Daha sonra programda düzenlemeler yapmak istediğinizde ben bu komutu niye kullanmışım veya burada ne yapmışım gibi sorulara cevap olarak açıklamaları okuyabilirsiniz.

MATEMATİKSEL İFADELER

C64ASM derleyicisi için aşağıdaki matematiksel işlemleri yazacağınız programlarda kullanabilirsiniz.

İSİM : Sembolik bir sabitin, değişkenin veya etiketin değeri programda kullanılır.
SAYI : 0 ila 65535 arasında bir onlu sayı (0 ve 65535 dahildir)
$SAYI : 0 ila $FFFF arasında bir Hex sayı
%SAYI : İkili sistemde bir sayı (max. %11111111)
"CHAR" : Herhangi bir ASCII karakter
'CHAR' : Karakter COMMODORE PETSCII olarak çevrilir.

* : **(Çarpı işareti)** Program Counter'in o anki değeri.Bu komutla programın başlangıç adresini tanımlarsınız.
Bu komutu isterseniz birden fazla kullanabilirsiniz. Daha iyi anlaşılması için aşağıdaki örneklere bakabilirsiniz.

```
*       = $0900          ; Programın başlangıç adresi olarak $0900 adresi tanımlanmıştır.
*       = 49152          ;     "     "     "     "  49152  "     "

        JMP   *          ; Bu komutla bilgisayar sonsuz döngüye girer

*       = $1000          ; Programın başlangıç adresi $1000 (4096)

        LDA   TABLO      ; Komut
        .....            ; Komut
        .....            ; Komut
*       = $4000          ; Tablonun başlangıç adresi $4000 (16384)
TABLO   .BYTE  0,1,12,33
        .byte  109,22,4,59
```

Aşağıdaki işlemler program yazarken sık sık kullanacağınız işlemlerdir.

< ; İki baytlık bir değişkenin değerinin veya bir tablonun, yazının bulunduğu adresin veya program içindeki bir yerin alçak baytı.
> ; Yukarıdaki açıklamanın yüksek baytı için olan şekli.

```
        ÖRNEK 1:                ÖRNEK 2:                ÖRNEK 3:
        sei                     lda   #<yazı            lda   #<tablo
        lda   #<irq             ldy   #>yazı            ldx   #>tablo
        ldx   #>irq             jsr   $ab1e             sta   store
        sta   $0314             .....                   stx   store+1
        stx   $0315             .....                   .....
        cli                     .....
        jmp   *                 rts
irq     inc   $d019     yazı    .text "c64"     tablo   =$3800
        lda   $d012             .byte 0         store   .byte 0, 0
        .....                   .....                   .....
```

Şimdi sırayla örnekleri açıklayalım:

İlk örnekte **"irq"** etiketiyle başlayan program bölümünün adresinin "alçak" baytı **"lda #<irq"** komutuyla okunuyor ve **"sta $0314"** komutuyla ilgili adrese yazılıyor. Aynı şekilde **"irq"** 'nın yüksek baytı için **ldx** ve **stx** komutları kullanılıyor.
İkinci örnekte benzer bir şekilde **"c64"** yazısı için yapılıyor fakat bir "**jsr"** komutu için kullanılıyor.
Son örnek biraz farklıdır. Bu sefer **"tablo"** 'nun adresi değil, tablonun değeri olarak tanımlanan bir sayı **A** ve **X** registerleri tarafından LOW-HIGH bayt şeklinde okunup (A=$00 ve X=$38'dir) **"store"** nin olduğu adreslere kopyalanıyor. Program yazarken bu şekildeki yazım şekillerini sık sık kullanacaksınız.

+ ; Toplama işlemi
- ; Çıkarma işlemi
* ; Çarpma işlemi
/ ; Bölme işlemi

ÖZEL KOMUTLAR : Bu bölümde assembler komutları olmayan bazı özel komutları açıklayacağım.

.BYTE : (BYTE LİSTESİ) Bu komut ile program içinde kullanacağız değişkenler için gerekli değerleri tanımlayabilirsiniz.

.WORD : .BYTE komutu gibidir. Fakat tanımlanan değer hafızada LOW BYTE - HIGH BYTE olarak saklanır.

.ASC "DİZİ" : Tırnak içinde verilen ifadenin karakterlerinin ASCII kodları hafızaya yerleştirilir.

.TEXT "DİZİ": .ASC komutu gibidir, fakat karakterlerin kodu CBM-ASCII'ye çevrilir.

.SCRL "DİZİ" : .TEXT komutu gibidir, fakat karakterlerin kodu "küçük harf / büyük harf " karakter seti kullanıldığı zamanki C64 ekran koduna çevrilir.

.SCRU "DİZİ" : .SCRL komutu gibidir, fakat karakterlerin kodu "büyük harf / grafik " karakter seti kullanıldığı zamanki C64 ekran koduna çevrilir.

Birazdan açıklayacağım komutların çok işinize yarayacağından eminim.

.INCLUDE asmdosyası : Daha önce yazmış olduğunuz .ASM uzantılı dosyaları bu komut ile yazmakta olduğunuz programa dahil edebilirsiniz. Fakat bir şeye dikkat edin. .INCLUDE komutuyla yüklediğiniz dosya ile yazmakta olduğunuz programda aynı isimlerin OLMAMASI gerekir. Mesela yazmakta olduğunuz programda TEMP isimli bir etiket olsun. Eğer yüklediğiniz dosyada da TEMP varsa C64ASM derleme sırasında hata verecektir. Ayrıca .INCLUDE ettiğiniz dosyaların yazmakta olduğunuz programdaki yerlerine dikkat etmeniz gerekmektedir.

.INCBIN dosyaadı : Bu komut ile daha önceden derlenerek oluşturulmuş olan bir **PRG, T64 veya P00** uzantılı bir dosya, program derlenirken eklenir.

.END : Bu komut ise yazdığınız programın sona erdiğini C64ASM programına bildirir.

Buraya kadar açıkladığım komutların haricinde yazmamış olduğum komutlar ve aritmetiksel işlemler bulunmaktadır. Eğer ingilizce biliyorsanız programı internetten indirip, klasörde bulunan c64asm.doc isimli dosyayı okuyabilirsiniz.

Bu sayının program köşesindeki program C64ASM ile yazılmıştır.

PROGRAM KÖŞESİ

```
;=====================================================================;
;                    TEXT ŞİFRE / DEŞİFRE PROGRAMI                    ;
;                          31-05-2003 / HADES                         ;
;=====================================================================;

                *=$0900

;---------------------------------------------------------------------;
mainsifre       jmp sifre
maindesifre     jmp desifre
;---------------------------------------------------------------------;
sifre           lda #<text
                ldx #>text
                sta read+1
                stx read+2
                lda #<sifrememory
                ldx #>sifrememory
                sta $fb
                stx $fc
                ldx #$00
                ldy #$00
looppack        jsr read
                asl
                asl
                sta temp
                inx
                jsr read
                jsr lsr4
                ora temp
                sta ($fb),y
                jsr read
                jsr asl4
                sta temp
                inx
                jsr read
                lsr
                lsr
                ora temp
                iny
                sta ($fb),y
                jsr read
                jsr asl6
                sta temp
                inx
                jsr read
                ora temp
                iny
                sta ($fb),y
                inx
                iny
                cpy length
```

```
                bne looppack
                rts
 ;-------------------------------------------------------------------;
desifre         lda #<sifrememory
                ldx #>sifrememory
                sta read+1
                stx read+2
                lda #<desifrememory
                ldx #>desifrememory
                sta $fb
                stx $fc
                ldx #$00
                ldy #$00
loopdepack      jsr read
                lsr
                lsr
                sta ($fb),y
                jsr read
                and #$03
                jsr asl4
                sta temp
                inx
                jsr read
                jsr lsr4
                ora temp
                iny
                sta ($fb),y
                jsr read
                and #$0f
                asl
                asl
                sta temp
                inx
                jsr read
                jsr lsr6
                ora temp
                iny
                sta ($fb),y
                jsr read
                and #$3f
                iny
                sta ($fb),y
                iny
                inx
                cpx length
                bne loopdepack
                rts
 ;-------------------------------------------------------------------;
lsr6            lsr
                lsr
lsr4            lsr
                lsr
                lsr
                lsr
                rts
 ;-------------------------------------------------------------------;

asl6            asl
```

```
                asl
asl4            asl
                asl
                asl
                asl
                rts
;-------------------------------------------------------------------;
read            .byte $bd,0,0
                rts
;-------------------------------------------------------------------;
length          .byte 3
temp            .byte 0
;===================================================================;
                *=$0a00
text            .scrl "c-64"    ; şifrelenecek text burada olacak
fuse            .byte 32,32,32

;===================================================================;
                *=$0b00         ; deşifre edilen text buraya yazılacak
desifrememory   .byte 0
;===================================================================;
                *=$0c00         ; şifrelenmiş text burada olacak
sifrememory     .byte 0
                .end
;===================================================================;
```

Bu sayının program köşesine hoş geldiniz. Bu programın hikayesi oldukça eskidir. Askerde teskere için gün sayarken, boş durmaktan sıkıldığım bir vakitte aklıma geldi ve kağıt üzerinde bitirdim. Daha sonra oturup kodladım ve diskete kaydettim. 1994'ten beri disketlerin birinde bekliyordu. Fakat bir zamanlar yapmaya başlayıp bıraktığım bir platform oyununda, ekran datalarını bu programlamla şıkıştırarak yer kazandım. Hatta yine başlayıp bıraktığım bir shoot-em-up oyununun HI-SCORE ekranında çıkacak isimleri bu programla şifreleyip daha sonra deşifre ederek ekrana yazdırıyordum. Aslında programın adı "TEXT PACKER-DEPACKER" idi. Fakat bazı sınırlamalardan dolayı "TEXT ŞİFRE-DEŞİFRE" adını koymayı uygun gördüm.

Lafı fazla uzatmadan programın teknik açıklamasına ufaktan bir giriş yapalım. Bilindiği gibi C64'te bir karakteri ekrana iki şekilde çıkarabilirsiniz. Birincisi doğrudan ekran belleğine karakterin - EKRAN KODUNU - POKE komutuyla yerleştirerek, ikincisi ise PRINT komutuyla tırnak işaretleri arasında yazarak C64'te ekrana bir yazı yazabiliriz..

İkincisi standart ASCII kodları kullanır ve C64 ile kullanabileceğiniz kodlar 32 ($20) ile 96 ($60) kodları arasındaki karakterlerdir. Bizim için önemli olan EKRAN KODLARI'dır ve sadece C64 için tasarlanmıştır. Standart ASCII kodlarda $40 ile $60 (64 - 96) arasındaki karakterler EKRAN KODU'nda 0 ile $20 (0 – 32) arasındadır, $20 ile $40 arasındaki kodlar ise değişmemiştir. Biraz daha ayrıntıya inersek bizim programın kullandığı ekran kodları -bizim işimize yarayacak olan ilk 64 karakterdir (0 - $40 arası)- 6 bitten oluştuğu için şifreleme veya sıkıştırma işleminin mantığını anlatmaya başlayabiliriz.

Yazının başında bahsettiğim sınırlama şifrelenecek yazının sadece ekran koduyla ve ekran kodlarınında sadece ilk 64 karakteriyle yazılmış olmasından kaynaklanmaktadır. İkinci bir sınırlama- aslında sınırlama değil mecburiyettir- şifrelenecek yazının 4 ve 4'ün katları uzunluğunda olmasının gerektiğidir.

Vereceğimiz örneğin basit olması için hafızada ekran kodlarıyla bulunan "ABCD" yazısını sıkıştıracağız veya şifreleyeceğiz. Bu yazı hafızada aşağıdaki şekilde bulunmaktadır.

ADRES (A HARFİ)							
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**

ADRES + 1 (B HARFİ)							
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**

ADRES + 2 (C HARFİ)							
0	**0**	**0**	**0**	**0**	**0**	**1**	**1**

ADRES + 3 (D HARFİ)							
0	**0**	**0**	**0**	**0**	**1**	**0**	**0**

Bize lazım olan ilk 6 bittir. Yani bizim kullanacağımız ekran kodlu bir karakterin genel yapısı şöyle olacaktır. → **0 0 x x x x x x** Bitlerin sıralanışı soldan sağa ve 7 6 5 4 3 2 1 0 şeklindedir. "ABCD" yazısını sıkıştırdığımızda 3 baytlık anlamsız bir yazı ortaya çıkacaktır. Şimdi diyeceksiniz ki, "Bu kadar işlemi sadece 1 bayt için mi yapacağız?" Ben de evet diyeceğim. Bu program temel programdır ve ayarlanınca isterseniz bütün hafızayı sıkıştırabilirsiniz. Şu anki yapısıyla programda geçen "length" değişkenini 192 yaparsanız 1 blokluk (256 baytlık) ve ekran kodlarıyla yazılmış bir yazıyı rahatça şifreleyebilirsiniz. Netekim, yazının başında bir oyun için kullandığımı yazmıştım. Ekran belleği 1 Kbayt'tır ve bu rutine bir iki ekleme yaparak 768 bayt olarak kısaltabilirsiniz. İşi biraz daha büyütürsek 20 ayrı ekran görüntüsünü hafızada sakladığınızı varsayalım. 20 ekran = 20 Kbyte eder. Bu programı kullanarak 20 ekranın kapladığı alan 16 Kbyte'a iner. Alın size 4 Kbyte'lık tasarruf. Sadece her 4 bayt'ta 1 bayt tasarruf ile nerelere geldik. Tabii birde şöyle bir soru var. Ben bu programı sadece ekran için mi kullanacağım? Hayır. Mesela bir program yazdınız ve programa kendi adınızı vs. info olarak yerleştirdiniz. Programınız çalışınca adınız çıkacak daha sonra program kullanılacak. Ama bazı lamer'lar sizin adınızı kendi adıyla değiştirerek ortaya çıkabilirler. İşte bu program ile adınızı sonsuza dek koruyabilirsiniz. Adınız, bilgileriniz şifreli bir şekilde programın herhangi bir yerinde duracaktır ve program çalışınca deşifre rutini devreye girerek adınızı vs. şifresiz şekilde ekrana yazacaktır. Kullanma alanı tamamen size kalmıştır ve konuyu daha fazla dağıtmadan programın açıklamasına devam edelim.

En son bize lazım olan ilk 6 bit demiştik. Yaptığımız işlemler şöyle. 1. harf xxx, 2. harf yyy, 3. harf vvv, 4. harf zzz ile gösterilsin. Ayrıca işlem sonunda şifreli sonucu saklayacağımız adreslerede SAKLA0, SAKLA1, SAKLA2 olarak adlandıralım. Birde geçici işlemler için TEMP isimli adres olsun.

1 – 1. harfi 2 kez sola kaydırın ve TEMP'e yazın **: 00xxxxxx** şu hale gelir → **xxxxxx00** → **TEMP'e**
2 – 2. harfi 4 kez sağa kaydırın ve sonuca TEMP ile OR işlemi uygulayın ve sonucu SAKLA0'a yazın **: 00yyyyyy** şu hale gelir → **000000yy** → **OR** işleminden sonra sonuç **xxxxxxyy** → **SAKLA0'a**
3 – İkinci harfi 4 kez sola kaydırın ve sonucu TEMP'e yazın **: 00yyyyyy** şu hale gelir → **yyyy0000** → **TEMP'e**
4 – 3. harfi 2 kez sağa kaydırın ve TEMP ile OR işlemi yapın ve sonucu SAKLA1'e yazın **: 00vvvvvv** şu hale gelir → **0000vvvv** → **OR** işleminden sonra sonuç **yyyyvvvv** → **SAKLA1'e**
5 – 3. harfi 6 kez sola kaydırın ve sonucu TEMP'e yazın **: 00vvvvvv** şu hale gelir **vv000000** → **TEMP'e**
6 – Son harfi ile TEMP arasında OR işlemi yapın ve sonucu SAKLA2'ye yazın : **vv000000** → **OR TEMP** → sonuç **vvzzzzzz** → **SAKLA2'ye**

Gördüğünüz gibi bütün işlem bu kadar. İşlem sonunda hafızada 00xxxxxx, 00yyyyyy, 00vvvvvv ve 00zzzzzz olarak sıralanan 4 harflik bir yazı (sayı da olabilir) "xxxxxxyy yyyyvvvv vvzzzzz" şeklinde şifrelenmiş veya sıkıştırılmış olur.

Deşifre programı ise bunun tam tersi mantıkla çalışır. Programı inceleyerek ne gibi işlemler yapılmış anlayabilirsiniz. Bu programı kullanmak için dikkat edilmesi gerekenleri tekrar yazıyorum. Öncelikle yazılarınız 0 ile 64 (64 hariç) arasındaki ekran kodları olmalı, yani grafik karakterler, inverse harfler olursa işinize yaramaz. İkinci konu ise yazınız mutlaka 4 ve 4'ün katları uzunluğunda olmalıdır. Peki uzunluğa dikkat etmezsek ne olur. Mesela 20 değilde 22 harflik bir yazınız var. Şifreleme sırasında 22'den sonraki karakterlerin yerine hafızadan rastgele bir değer okunacak ve şifreleme yapılacaktır. Ama deşifre sırasında rastgele okunan baytlar deşifre sırasında yazıyla ilgisi olmayan birkaç harf şeklinde ekranda sırıtacaktır. Bunun çaresi ise yazınızın sonuna boşluk karakteri ekleyerek uzunluğu 4'ün katları haline getirmektir.

Programın başlangıç adresi $0900 (2304)'te bir sıçrama tablosu vardır. Eğer BASIC'ten SYS 2304 yazarsanız şifreleme bölümünü, SYS 2307 yazarsanız deşifre bölümünü çalıştırabilirsiniz. Assembler bilenler için bir sorun olacağını zannetmem. Veya rutinleri ayırıp ayrı ayrı kullanabilirsiniz. Umarım işinize yarar.

Eğer bu program ile ilgili sorularınız olursa aşağıdaki e-mail adreslerine bir e-mail atın yeter.

hades6510@mynet.com ve hades6510@yahoo.com

AEGİS/BRONX RÖPORTAJI

Bu sayıdan itibaren eğer bir aksaklık olmazsa her sayıda bir C64 fanatiği ile yapılan röportaja yer vereceğiz. İlk röportajı uzun zamandır C64 ve AMIGA kullanmakta olan **AEGİS** ile yaptık.

HADES : Selam AEGİS, röportajımıza başlamadan önce röportaj isteğimizi kabul ettiğiniz için teşekkür ederiz.
AEGİS : Birşey değil.

HADES : Bize gerçek adınız, yaşadığınız yer, doğum tarihiniz, işiniz vs., yani kendinizden bahseder misiniz?
AEGİS : Adım Timur, İstanbul/Türkiye'de yaşıyorum, 24.02.1976 doğumluğum, özel bir şirkette Bilgisayar ve barkod sistemleri üzerine çalışıyorum. Evliyim ve bir de kızım var.

HADES : Genel anlamda ne zamandır bilgisayar kullanıyorsunuz? İlk bilgisayarınızı ne zaman aldınız ve nasıl bir bilgisayardı?
AEGİS : İlk bilgisayarımı ortaokul yıllarında 88 senesinde almıştım ve ilk bilgisayarım Commodore 64 oldu tabii ki. Hatırlıyorum da babam taksitle almıştı 1500 TL felandı sanırım. O zamandan bu zamana kadar halen C64 kullanırım fırsat buldukça.

HADES : Bildiğiniz gibi C64 TÜRKİYE dergisi sadece COMMODORE 64 için ve online olarak yayın hayatına başlamış durumda. Dolayısıyla röportajımız daha çok C64 ağırlıklı olacaktır. İşte size çok kısa bir soru: SİZCE C64 NEDİR?
AEGİS : Bence Commodore 64 bir efsane. Bir çok bilgisayarın atası diyebilirim bence.

HADES : C64 ile ne zamandır uğraşıyorsunuz ve ciddi anlamda uğraşmadan önce C64 ile neler yapıyordunuz?
AEGİS : 88 senesinden beri uğraşıyorum o zamanlar daha çok oyun ağırlıklı idi. 90 senesinde makina diline merak sardım o zaman elimden bir çok kartuş geçmişti (Expert, Final 3, ICE Machine). 91 senesinde scene olayını keşfettim. İlk kodumu Multi Ice 3 kartuşunun makina dili editöründe yazmıştım. Hahahaaa... ne günlerdi ama. Sonra swap dediğimiz yurt dışındaki arkadaşlarla hem yazışma hem de disk değiş dokuş yapma olayına girişmiştim. Bu sayede bir çok arkadaşım olmuştu. Tabii aradan uzun zaman geçti. Halen ararım o günleri. İnternet çıktı yiğitlik bozuldu. ☹

HADES : C64 dünyasında bireysel olarak bir şeyler üretmek mümkün, fakat birkaç kişi bir araya gelip bir grup kurabilir ve daha çabuk birşeyler üretebilir. Bir grup kurulabilmesi için neler gereklidir ?
AEGİS : Öncelikle TAKIM ruhu gerekiyor. Bu bence çok önemli. Ondan sonra gerisi geliyor zaten. Friendship Rulaz!!! ☺

HADES : Herhangi bir grupta bulundunuz mu veya bir grup kurdunuz mu? Gruptaki göreviniz neydi? Bireysel veya grup olarak neler yaptınız? Şu anda bir grupta mısınız?
AEGİS : Birçok grupta bulundum. Ex-Atlantis, Ex-Excess, Ex-Axelerate ve benim eski grubum Caisson (Can) ile kurduğumuz bir çok ilke imza attığımız ASCRAEUS grubu idi. Şimdi efsane grup BRONX'un üyesiyim.

HADES : Anladığımız kadarıyla epeyce eski bir C64'çü sayılırsınız. Türkiye'deki C64 scene tarihi hakkında neler söyleyebilirsiniz? Bugün için neler söyleyebilirsiniz ?
AEGİS : Scene olayı oldukça iyidi ben girdiğim senelerde köklü bir yapısı vardı. Bir sürü grup vardı örnek verecek olursak Medal, Zombie Boys, The Jocker Crew, Clique vb. Ancak olay 91 senesi ve tabii ki Amiga bilgisayarlar piyasa çıkınca Türkiye için scene olayı bitmiş duruma geldi ancak bitmedi tabii ki. Şimdi ise 64 kullanıcıları parmakla gösterilecek kadar azdır. Ama fanatikleri oldukça fazla. Biz C64 efsanesini sürdürmeye kararlıyız.

HADES : Bildiğimiz kadarı ile bir zamanlar C64 SCENE'i oldukça hareketliydi. Sürekli olarak demolar, disk mag'ler, tools disk'ler dağıtılır, oylamalar yapılır, özellikle yurt dışında çeşitli partiler, yarışmalar düzenlenirdi. Şu anda durum nasıl? Türkiye'deki durum nasıl? Scene'de olup bitenleri takip edebiliyor musunuz?

AEGİS : O zamandan bu zamana çok şey değişti tabii ki. Ama hız aynı sanırım parti olayları halen devam etmekte. Mesela Türkiye'de 2002 baharında 7D2 partisi yapılmıştı. Scene az da olsa vakit buldukça takip etmeye çalışıyorum. Disk magler, demolar eskisi gibi çıkmaya devam ediyor. Yakında Türkiye'de 7D3 Bronx grubunun organize ettiği bilgisayar partisi yapılacak. Bu arada tabii ki C64 ile haberleri güncel olan www.c64.sk sitesi.

HADES : C64'te kendinizi yeterli buluyor musunuz? Şu veya bu konuda daha çok çalışmam gerekiyor diye düşünüyor musunuz?

AEGİS : Hayır. Yeterli bulmuyorum tabii ki. Öğrenecek daha çok şey var ama vakit olayı yüzünden fırsat bulup da C64 bir yana dursun, PC ile uğraşamıyorum.

HADES : Scene'de favorileriniz nelerdir? Özellikle beğendiğiniz veya size göre her zaman 1 numara olan grup, demo, müzik vs var mı?

AEGİS : Benim beğendiğim coderlardan Türkiye'de bizim eski gruptaki arkadaşım Hades "İsmail Şahin", Madhead "Levent Delibaş"/Ascreaus, Wisdom "Hüseyin Kılıç"/Crescent, Skate "Emir Akaydın"/Bronx, yutdışında ise Aeg, naphalm, Demo olarak Mathamatica, Dutch Breze, Dawnfall müzisyen olarak Pri, Mitch&Dane, Rob Hub., grup olarak Excess, Atlantis, Oxyron daha var ancak aklıma şu an için gelmiyor.

HADES : Benim kişisel görüşüme göre Türkiye'de C64 scene'i nerdeyse bitmiş durumda, fakat yurt dışında hala partiler yarışmalar vs. düzenleniyor. Çok kaliteli internet siteleri var ve oradakiler bu makinanın kıymetini biliyorlar. Bu konuda ülkemizdeki C64'çüleri tekrar aktif hale getirmek için genel olarak neler yapılabilir? Siz neler yapabilirsiniz? Yoksa sadece yurt dışındaki gelişmeleri internetten takip edip tüketici mi olacağız? Sizce ülkemizdeki C64 scene canlanır mı?

AEGİS : Gerçeği yazmak gerekirse scene ülkemizde canlı tutmak çok zor olsa gerek. Yeni neslin Commodore 64 ve eski bilgisayarlardan pek haberi olmadığı için biz son C64 kuşağındanız diyebilirim. Ben efsaneleri yaşatmak için internette bir site açtım. Adı Amiga Türk (www.amigaturk.net) Commodore 64 ve Amiga ağırlıklı bir site. Bunun benzeri bir kaç C64 fan siteleri de mevcut. Sanırım tüketici olacağız gibi görünüyor.

HADES : Yurt dışında oldukça hareketli olan C64 scene'in yanısıra bir zamanlar belki de hayal olan gelişmeler var. Bunlardan biri de Amerikalı C64'çü bir bayanın hobi olarak başladığı C64 için bir video hızlandırıcı kart projesinin COMMODORE ONE projesine dönüşmesidir. Halen geliştirme aşamasında olan C-ONE hakkındaki görüşlerinizi öğrenmek istiyoruz. C64 ile harika şeyler ortaya çıkarıldığına göre bu makine ile neler yapılabileceğini tahmin edebiliyor musunuz? C-ONE ile ülkemizdeki scener'ler tekrar aktif hale gelebilir mi?

AEGİS : Sanmıyorum. Cihaz halen prototip aşamasında. Devamlı revizyona uğruyor, peki alsak bile Türkiye'ye sağlam olarak ulaşması zor gibi geliyor bana. Ve herkese hitap etmiyor C-ONE. Biz son kuşağa hitap eden bir makina olarak görüyorum. Aktif hale getirmesi tartışılır ama bence oldukça zor.

HADES : Son olarak bizim aklımıza gelmeyen fakat sizin söylemek istedikleriniz var mı ve C64 TÜRKİYE için bir şeyler yapmak ister misiniz?

AEGİS : C64 Türkiye bence uzun zamandır olması gereken bir projeydi. Sizi tekrar tebrik etmek isterim. Derginizde olması gereken bence bir hardware köşesi açmanız. Başarılarınızın devamını dilerim. Bol C64'lü nice güzel günler dileğiyle.