# Plazma 5

# Plazma 5

Yayımlanma 2008

Bu yayında verilen bilgiler, faydalı olması umuduyla hiçbir garanti dahilinde olmaksızın verilmiştir. Bir okuyucunun buradaki bilgileri kullanarak kendisinin veya bir başkasının yazılım, donanım veya verisine zarar vermesi halinde, yazarlar veya dergi editörleri, hiçbir şekilde sorumlu tutulamaz.

Yayınlanan bütün yazıların her hakkı yazarlarında saklıdır. Plazma dergisi bu yazıları, süresiz olarak, çıktığı bütün formatlarda yayınlayabilme iznini almıştır. Yazıların hakları ile ilgili yazarlarla bireysel bağlantıya geçilebilir.

# İçindekiler

# Editörden

Şu an iki ayrı sebepten dolayı iç rahatlığı yaşıyorum.

Birinci ve daha göz önünde olan sebep, Plazma'nın 5. sayısını, öngördüğümüz sürede ve öngördüğümüzden iyi bir içerik kalitesiyle çıkarabilmiş olmamız. Bir önceki sayının ilk hedefimizden birkaç hafta sapması, aslında bu sayı için ayırdığımız süreyi de kısıtlamıştı. Neyse ki Plazma ekibindeki bütün yazar ve editör arkadaşlar, amatör ruhlarına rağmen, bu sayı ile ilgili hazırlıklarında kelimenin tam anlamıyla "profesyönel" bir iş çıkardılar. Kısacası Plazma ekibi olarak sözümüzde durduk ve siz okuyucularımıza, yine dolu dolu bir sayıyı tam vaktinde getiriyoruz.

İkinci sebep ise biraz daha arka planda olan gelişmelere bağlı. Derginin mutfağında kullandığımız teknik araç gereçlerle ilgili önemli gelişmeler kaydettik. Bunun sonucu olarak dergide yayınlanan yazıların Docbook formatına çevrilmeleri ve bu Docbook kaynak yazılarının da daha sonra html ve pdf versiyonlarına otomatik çevrilmesinde, artık sürecin neredeyse tamamı otomatize edilebildi. Bu sayede bu basamaklarda elle düzeltmeler yapmaktan kurtulmuş olduk. Dolayısıyla, hem olası hatalar azalmış oluyor hem de zaman kazanmış oluyoruz. Tabii herşey güllük gülistanlık da sayılmaz. Örneğin, derginin pdf versiyonunda, 4.sayıda da olduğu gibi, pekçok yerde satır sonuna gelen kelimelerde hecelerin doğru bölünmediğini göreceksiniz. Bu, Docbook'tan pdf'e yapılan otomatik çevrim sırasında oluşan ve henüz çözemediğimiz teknik bir hata. Fakat yinede 4. sayıya göre, arka plandaki otomasyon süreçlerimizde yaşanan bu ilerleme, sonraki sayılar için benim içimi rahatlatmakta rol oynuyor.

Bu sefer editörden yazısında tek tek yazılardan bahsetmeyeceğim. Bu bilgi için sizleri "Plazma'nın Bu Sayısında" başlıklı yazıya davet ediyorum. Göreceğiniz gibi zaten güçlü olan yazar kadromuzu, yeni eklemeler ve misafir yazarlarla daha da güçlendirdik. Kapsadığımız platform ve teknolojilere yenilerini eklemeye devam ediyoruz. Eğitici yazı dizilerimiz çoğalıyor. Amatör bilgisayar dünyasının sosyal boyutunu irdelemeye de devam ediyoruz. Kısacası, hiçbir ticari gelirimizin, giderimizin veya kaygımızın olmamasının avantajını sonuna kadar kullanıyor ve size ticari dergilerin getirmekte zorlanacağı yazıları ulaştırıyoruz. Bu yazılarla "bu dergi satmaz" veya "reklam alamayız" gibi şeyler düşünmek zorunda olmama lüksümüz var :)

Asıl bu köşede yazmam gereken daha önemli bir konu, daha doğrusu etmem gereken büyük bir teşekkür var. Plazma'nın 4. sayısı pekçok yönden beni çok çok mutlu ettiyse de dergide gözlemlediğim önemli bir eksik vardı. Bu eksik, dergide olması gerektiği kadar haber ve güncel yazı olmamasıydı. Kimi insanlar artık internet çağında haberlere gerek olmadığını savunuyor. Bu eğer doğru olsaydı dünyadaki gazeteler yok olurdu. Günümüzde internet herkesin elinin altındayken, iyi ve ilgi çekici bir haber köşesi hazırlamak artık daha zor, bu doğru. İnternetten haberleri kopyalayıp yapıştırırsanız çok da fazla bir değer yaratmış olmuyorsunuz. Günümüzde iyi haber yazıları için farklı kaynaklardan araştırmak ve veri derlemek gerekiyor. Her haberin cevaplaması gereken 5N1K sorularının dışında, yorumlarla da zenginleştirilmesi gerekiyor. Bütün bunlar yapıldığında, okuyucu için artı değer yaratan kaliteli haber ve inceleme yazıları yaratılabiliyor.

İşte bu çok efor isteyen büyük iş için dergide yeni bir yardımcı editör görevlendirilmesi gerektiğine karar verdim. Bunun sonucu olarak teklif götürdüğüm Skate, bu görevi tereddütsüz kabul ettiği gibi ilerleyen sayfalarda göreceğiniz üzere, bu konuda inanılmaz iyi bir iş çıkardı. Plazma'nın bu sayısında Skate sayesinde, bana göre "Bomba" gibi bir "haber ve güncel yazı" içeriğimiz var. Bu noktada bize bazı güncel yazıları hazırlayan konuk yazarlarımıza da teşekkür ediyorum. Tabii haber editörlüğündeki azmi ve başarısından dolayı da burada Skate'e birkez daha tebrik ve teşekkürlerimi sunuyorum.

Daha fazla kafanızı şişirmeden, sizi derginizin yeni sayısı ile başbaşa bırakıyorum. Keyfini çıkarın...

# Plazma'nın Bu Sayısında...

## Güncel Yazılar

### Amatör Bilgisayar Dünyasından Haberler

Cevval haber departmanımız, gece demedi gündüz demedi, sizler için amatör bilgisayar aleminde kıyıda köşede ne kadar ilginç haber varsa topladı.

### Z80 Programlamada Yeni Ufuklar "CCZ80 Dili"

8 bit dünyasının bu kadim işlemcisi için, günümüz olanaklarından faydalanarak program yazmanın keyfini ve konforunu yaşamak isteyenler hemen klavye başına! Bizden tanıştırması...

### C64TPC – Proje Günlüğü

Türkiye'den hergün amatör donanım projeleri çıkmıyor. Bu yüzden C64TPC gibi ilginç bir proje ortaya çıktığında hemen gidip yapanlara ulaşıyoruz. Bu çok özel yazıda, proje boyunca karşılaşılan zorlukları ve yaratım sürecinin nasıl bir deneyim olduğunu birinci ağızdan, yani projeyi gerçekleştiren Ahmet Zeki Eymür'ün kaleminden okuyacaksınız.

### Natami'ye İlk Bakış

Son zamanlarda Amiga camiasını en çok heyecanlandıran projelerden biri Natami oldu. Ülkemizdeki en aktif retro cihaz koleksiyoncularından LW3D, Plazma okuyucuları için Natami'yi mercek altına aldı

### Lorraine'den Minimig'e Amiga

Bu başlıkta sadece Amiga kelimesi size tanıdık geliyorsa çok şey kaçırıyorsunuz demektir :) Bu yazıda ilk Amiga prototipi olan Lorraine ile, FPGA üzerinde bir Amiga implementasyon projesi olan MiniMig'in geliştirilme süreçlerini, favori yazarlarımızdan CoZe ile beraber incelemeye ne dersiniz.

### MossyCon

Kuzey Amerikada Commodore kullanıcıları aktiviteye devam ediyor. Muhabiriniz Nightlord hiçbir masraftan kaçınmayıp (!) sizler için Oregon'dan bildiriyor.

### Forever 9

Doğu Avrupa'nın gözde 8-bit partilerinden Forever, konuk yazarımız Jailbird'ün gözünden siber alemde mesafeleri aşıp Plazma okuyucularının monitörlerine geliyor.

### Rakı Balık Spectrum ve Breakpoint 2008

Dev bütçeli (!) bir yapım olan Plazma Derginiz, dev bütçeli diğer bir yapım olan Breakpoint partisine Spectrum yazarımız olan Ref'i gönderdi. Hiçbir masraftan kaçınmadık sayın okuyucular. Alman Başkonsolosluğu'nun özel davetlisi olarak Sheraton Bingen Oteli'nde kalan Ref, süitinden Breakpoint'teki gelişmeleri sıcağı sıcağına aktardı.

## Yazılım ve Donanım

### C++ Kursu 1

Yıllardır oyun yapmak istıyorsunuz. Fakat herkes size C++ öğrenmeniz gerektiğini söylüyor. Web'de ne kadar aradıysanız da Türkçe kaynak bulamadınız. Bulduğunuz kaynaklar da ileri seviyede. O zaman bu yazı dizisini takip edin

### Gnu Make ve Makefile Kullanımı

Make özellikle Unix(linux,bsd vs.) ortamlarında sıkça kullanılan bir otomatik derleme aracıdır. Bu derste make programının kullanımına giriş yapıp, basit bir makefile nasıl hazırlanır onu öğreneksiniz, hem de damarlarında Open Source kanı akan Ragnor'un kaleminden :) .

### Bilgisayar Mimarisine Giriş

Yıllar önce Nightlord tarafından Web'de yayınlanan ama sonra kaybolan iki bölümlük bu yazı, Plazma için yeniden editlenip birleştirildi. Bu sayıda basit bir işlemcinin nasıl çalıştığına ve bellek ile nasıl etkileştiğine değiniliyor.

### PHP Kursu 2

PHP yazısının ikinci bölümünde, 1. bölümde işlenen dilin temel özellikleri, değişkenler ve fonksiyonlar konularından sonra "sınıflar" konusu işlenerek PHP dilinin yapısı hakkında bilinmesi gereken en önemli konular sona eriyor. Bu aşamadan sonra PHP ile veritabanı uygulamaları konusuna giriş yapılıyor.

### Action Script 3 (Usta...)

Flash öğrenip çılgın ve janjanlı web uygulamaları yapmak için yanıp tutuşanlarınız var bunu biliyoruz. Bunu bildiğimiz için de size Spaztica'nın kaleminden süper bir yazı dizisi getirmeye başlıyoruz. Hem Flash öğrenin, hem Zen...

### 68000 Assembly ve AsmPro

Amiga assembly kursumuz son sürat devam ediyor. Kursun bu bölümünde Hardware Register'larının kısa açıklamaları, 68000 Assembly komutlarının açıklamaları (Dallanma komutları),

Örnek kodlar (Ascii2Decimal çevirici, copper ile ekran açma) sizleri bekliyor. Ayrıca AsmPro (ve AsmOne) Assembler editörlerinin kullanımı hakkında da kaynak bir yazı Plazma'da (ellerinizden öper)

## CoZe'nin Amiga Donanım Köşesi

Amigacı'ların vazgeçemediği köşelerden Coze'nin köşesinde bu sayıda Amiga kullancılarının çabalarıyla şekillenmekte olan çeşitli donanım projeleri hakkında Coze'nin izlenimlerini okuyacaksınız.

# Muzik

## Drey'in Sequencer Köşesi

Elektronik müzik meraklılarının sevgilisi olan bu bölümde geçen sayıda kaldıgımız yerden Reason programını incelemeye devam ediyoruz. Bu sayıda Subtractor' ı mercek altına yatırıyoruz. Analog bir synth olan Subtractor ile oldschool sounduna dönüş bizi bekliyor! Muzik gurumuz (ve gururumuz) Drey'in kaleminden...

# Alternatif Platformlar

## Amstrad CPC Demoscene

Alternatif platformları geçen sayıdan itibaren kapsama alanına alan Plazma, Amstrad CPC yazarımız Alcofribas'ın usta kaleminden, CPC demoscene'e tanıtmaya başlayan harika bir yazıyı daha size ulaştırıyor. İleriki sayılara temel teşkil edecek bu başlangıç yazımızda; CPC dünyası için kilometre taşı olmuş kişi, ülke, disk dergisi ve demoları kendi bakış açımızla sizlere de tanıtmaya çalışacağız.

## Atari 800 XL

Platform kapsamımızı sürekli büyüteceğimizi söylemiştik. Araştırmacı gazeteciliğin kitabını yeniden yazan Skate'in kaleminden, 8-bit bilgisayar ailesinin en eski fertlerinden biri olan Atari 800XL üzerinde bir inceleme yazısı bu sayıda karşınızda. Bilgisayarın çeşitli donanımsal özelliklerine ve hafıza adreslerine genel bir bakış attıktan sonra Atari 800XL için nasıl Assembler programları hazırlanabileceği konusuna değiniliyor. Yani bu yazıyı okuyup biraz çalışarak Atari 800 XL'iniz için oyun ve demolar kodlamaya hemen başlayabilirsiniz

# Sosyal Yazılar

## Bir Scener Eşi'nin Gözünden Demoscene ve Scenerlar

Yıllardır bu obsesyondan çok çeken scener eşleri (kız/erkek arkadaşları ve hatta anneler, babalar, kardeş ve kuzenler) bu yazıda kendinizi bulacaksınız. Bu “scener” denilen adamlar manyak mı? Bu kadar gün ve saat bu makinelerin başında ne yapıyorlar? Odalarından sürekli gelen ve sizin başınızı ağrıtan bu bip bip sesler nedir? Nedir? Nedendir? Sizin gibi bu cevapları arayan bir diğer scene kurbanı Irian'ın kaleminden...

## Coder/Sanatçı olmak üzerine

İlk bakışta birbirinden çok farklı disiplinler gibi görünen bu iki rol arasındaki felsefi benzerlik ve ilişkilerin izini sürmek ilginizi çeker miydi? İçinizdeki filozofu Spaztica'nın bu ilginç yazısı ile beslemeye ne dersiniz?

## Türk Scene Tarihi – 2

Geçen sayıda başlayan ve okuyuculardan süper tepki alan bu yazı dizisi, bu sayıda da son sürat devam ediyor. Vigo'nun dipsiz kuyu gibi olan hafızasından ve esprili kaleminden bu sayıda C64 platformundaki Türk gruplarının yaptığı Demo çalışmalarının tarihçesini okuyabilirsiniz. Daha önce olduğu gibi şimdi de bütün eski Türk scenerlardan bu tarihi zenginleştirmek ve kayıt altına almak için yardım bekliyoruz. Kendi anılarınızı paylaşmak veya olası ekleme ve düzeltmeleriniz için bize ulaşın.

# Amatör Bilgisayar Dünyasından Haberler

## Haber Editöründen

Merhaba,

Bu sayıda sevgili editörümüz Nightlord'un talebi üzerine haber editörlüğü görevini üstlendim. Aslında haberler açısından çok da kötü bir döneme denk gelmedim, hemen her platformda yeni çıkan donanımlar, Breakpoint gibi devasa bir scene partisi, diğer irili ufaklı partiler, yayınlanan demolar, grafikler, müzikler, spekülasyonlar, kısacası hemen herşeyden biraz mevcuttu. Olayın haber çöplüğüne dönüşmemesi için elimden geldiğince ilgi çekici başlıkları toparlamaya çalıştım. Aralarından önemli gördüğüm başlıkların detaylandırılması için çeşitli yazarlarımızdan destek aldım ya da şahsen yazdım.

Bu haber yazıları, kısa kısa haber başlıklarının yanı sıra inceleme yazıları ve parti raporlarını da kapsıyordu. En ilginçlerinden biri Jailbird / Booze Design'dan gelen Forever 9 parti raporu oldu diyebilirim. Çok eğlenceli bir yazı ve hemen hemen partiyle ilgili hiçbirşey içermiyor. ;) Ayrıca Hydrogen, Endo ve benim tarafımdan yazılan Cauldron demosunun inceleme yazısı da yıllar önce Bronx adına çıkardığımız 64 Times isimli derginin ruhunu 7-8 sene sonra yeniden yakaladı diyebilirim. Dolayısıyla ciddi haberlerin yanı sıra sizleri biraz tebessüm ettirecek, hatta zaman zaman güldürecek içerikler de sizleri bekliyor.

Derginin çıkmasına yakın gelişen olaylar ve yakın zamanda sona ermiş partiler ile ilgili daha detaylı bilgileri bir sonraki Plazma'da bulmanız mümkün olacaktır.

Emir 'Skate' Akaydın

# Bilgisayar Organizasyonları

## 2008 Yılı İçersinde Gerçekleşen Organizasyonlar:

1. C64.sk Cover Compo #1 - 1-31 Ocak 2008
2. Extaz 2008 - 1-3 Şubat 2008 Raww.orgy 2008 – 15-17 Şubat 2008
3. Kbit 2008 - 22-24 Şubat 2008
4. Oxyron Party 2008 - 22-24 Şubat 2008
5. Beach Party '08 - 23 Şubat 2008
6. En MAZZA Data 3 - 7-9 Mart 2008
7. Insiders Meeting 2008 - 10 Mart 2008
8. Forever 2008 - 14-16 Mart 2008
9. Bra Data 2.0 - 15-16 Mart 2008
10. Breakpoint 2008 - 21-24 Mart 2008

## 2008 Yılı İçersinde Gerçekleşecek Organizasyonlar:

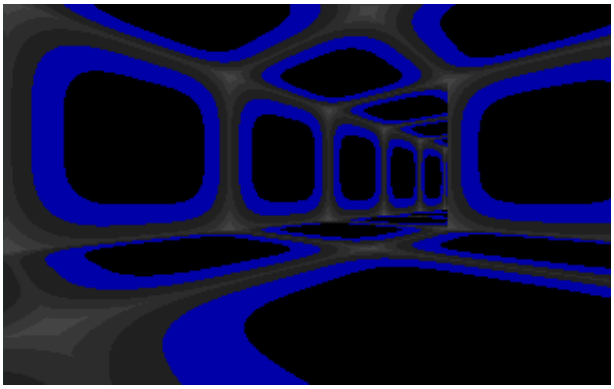
1. Rewired 2008 - 20-22 Haziran 2008
2. Back in Time Live Stockholm 2008 - 13 Eylül 2008
3. X'2008 - 24-26 Ekim 2008
4. The Last Compo (TM) - 1-24 Kasım 2008 (C64.sk SID compo #8)

* Breakpoint partisine 4 Türk scener katılım gösterdi. Decipher, Impetigo, Ref ve Wisdom.

# PC dünyası

1. Bronx grubu yaptıkları mini bir buluşmanın ardından aSSmaLL mEEtro isimli bir buluşma introsu yayınladı. http://www.pouet.net/prod.php?which=50210
2. Breakpoint partisini bu sene bir parti davetiyesi olan "Masagin - Nvision 08 Invitation" isimli ürünleriyle Farbrausch ve Neuro grupları kazandı. Andromeda Software Development'ın tamamen 2 boyutlu olarak hazırlanmış "Metamorphosis" isimli demosu ikinci olurken Cocoon'un "Shad 3" isimli yüksek kalitede rendera sahip sahnelerden oluşan demosu 3.lüğü aldı.
3. Breakpoint partisinin en çok ses getiren ve tartışılan olaylarından biri Plastic grubunun Sony Entertainment desteğiyle Play Station 3 platformu için hazırladığı "Linger In Shadows" demosu oldu. Yarışmaya katılmayan ve ticari bir ürün olduğu için tam bir scene demosu sayılmayan demonun kalitesi, birçok tartışmayı beraberinde getirdi.
4. (Nightlord: Bir haber de benden...) Breakpoint Partisinde olan ve Türkiye demoscene açısından önemli bir gelişme de "Rakı, Balık, Spectrum" köşesinin yazarı Ref'in Türkiye'nin eski ve en kaliteli gruplarından olan Crescent ekibine, demo/art director rolüyle katılması oldu. Sinema ve TV alanında çalışmaları olan Ref'in Wisdom ve İmpetigo gibi usta scenerlar ile güçlerini birleştirmesinin doğuracağı kaliteli sonuçları heyecanla bekliyor olacağız

## Neon Station / Digimind – 256b PC Intro



**Şekil 1. Neon Station / Digimind**

Fantastik bir 256 bytelık çalışma olan Neon Station yayınlandı. Herkes hala “nasıl oluyor da oluyor”u tartışıyor.

Bundan seneler önce PC scene’i ilk olarak “Tube” isminde 256 bytelık bir intro ile iç içe geçmiş sarmal tünellerin içersinde kamerayla dolanarak sarsılmıştı. Tube’den bu yana birçok etkileyici 2b/3b efekt hatta ses/müzik içeren 256 byte introlar yapıldı. Fakat Tube’un yarattığı etkiyi sanırım seneler sonra ilk kez “Neon Station” yaratabildi.

Neon Station ilk başta standart yer ve tavan dokusuna sahip bir raycasting motoruna benziyor. Ancak kameranın alt kata inip, yeniden üst kata çıkarak dairesel dolanım yapması işleri değiştiriyor. Şaşırtıcı faktörler şunlar:

1. Tüm bloklar tek bir texture kullanıyor ancak textureın şekli ve renkleri oldukça kaliteli gözüküyor. Bu kadar özelliğe sahip olan ve boyut problemi yaşaması muhtemel bir intro için standart XOR texturelarının dışına çıkılmış olması dikkatlerden kaçmıyor.
2. 256 byte için oldukça geniş bir harita üretilmiş, hem de birden fazla kata sahip bu harita. Harita her defasında aynı şekilde üretiliyor yani random bir faktör kullanılmamış. Bilinçli bir fonksiyon ile harita içinde dolaşılabilecek şekilde üretilmiş.
3. Kamera oldukça yumuşak sinüs hareketleriyle dolanıyor. Asıl öncemli nokta kameranın hareketleri sırasında hiçbir duvarın içinden geçmeden güzel bir şekilde boş alanlardan alt kata inip üst kata çıkarak dolanıyor olması. Kamera sabit bir yol izliyor. Sanırım haritadaki alanları oluşturan fonksiyon ile kamera ortak bazı kod parçaları kullanıyorlar.
4. Bütün bunların yanısıra ESC tuşu da çalışıyor.

Neon Station hakkında daha övgüyle bahsedilebilecek çok şey var. Ama hala izlemediyseniz izleyip kendi yorumunuzu yapmanızı öneriyorum. Neon Station Pouet linki: http://www.pouet.net/prod.php?which=49856

Digimind’ın (Neon Station’ın yapımcısı) diğer çalışmaları: http://www.pouet.net/groups.php?which=1941

# Amiga dünyası

1. Breakpoint partisini Drifters grubunun 20. yıllarının şerefine yayınladığı Twenty isimli demo kazandı. Ancak 2. olan Elude grubunun Soliloquy isimli demosunun teknik olarak daha üstün olması nedeniyle ödül töreni sırasında ilginç olaylar yaşandı. Öncelikle Elude grubuna oy veren kişiler Drifters’ın birinci olduğunu duyunca tepki gösterip yuhalamaya başladılar. Birincilik beklemeyen ve Elude grubunun demosunun kazanması gerektiği ile ilgili seyircilerle (özellikle yuhalayanlarla) hemfikir olan Drifters grubu ödülü reddederek, 2. olan Elude grubunun ödülü almasını sağladı.

## Minimig

9 Şubat 2008 tarihinde Acube firması tarafından satışa sunulan Minimig donanımının ilk üretilen serisi 1 aydan kısa bir süre içersinde, 7 Mart 2008 tarihinde tükendi.Minimig, “Mini” A”mig”a kelimelerinden türemiş bir isim. Mini derken gerçekten mini, 12x12 cm ebatlarındaki bir cihazdan söz ediyoruz. Minimig, Amiga 500’ün mimarisini esas alıyor. Ancak standart bir Amiga 500’den daha gelişmiş özellikler de içeriyor.

**Şekil 2. Minimig**

Özellikler:

1. İşlemci: 16Mhz MC68SEC000

1. RAM: 2Mbyte 45ns SRAM

2. FPGA(*): 400Kgate Spartan-3 (XC3S400)
3. Boot/sistem denetleyici: PIC18LF252
4. Depolama: MMC flash kart
5. Güç: +5VDC (mousesuz olarak yaklaşık 100mA)

Giriş/Çıkış:

1. Çift PS/2 port (mouse/klavye)
2. Çift 9-pin joystick port
3. VGA port
4. RS-232 port
5. Audio çıkışı

Diğer:

1. Jtag arabirimi, güç/sürücü ışıkları
2. Reset ve OSD butonu

*: Field Programmable Gate Array

Fiyat:

İlk açıklanan satış fiyatı 138 Euro

## Uyumluluk:

Minimig, Amiga 500 ile %100 uyumlu diyemeyiz. Şu ana kadar 100'ü aşkın oyunla yapılan testin sonucunda 30 civarı uyumsuz oyuna rastlandı. Yani Amiga 500 ile uyumluluk seviyesi şimdilik %70'de diyebiliriz. Testlerde başarısız olan oyunlardan bazıları şunlar:

1. Chaos Engine 1&2
2. Gods
3. Impossible Mission II
4. Bubba 'n' Stix
5. Airport
6. Alien Breed Special Edition
7. Alien Breed Tower Assault
8. Agony
9. Overdrive
10. First Samurai
11. Super Skidmarks
12. UGH!
13. Lionheart
14. Walker
15. Desert Strike
16. Skidmarks
17. Super Star Dust
18. Prince of Persia
19. Zool
20. Zaxxon
21. S.W.O.S
22. Wonder Dog
23. Utopia
24. Nemac IV
25. Torvak
26. Worms
27. Wolfchild
28. Tin Toy
29. Cannon Fodder 1&2
30. Cardiaxx
31. The Simpsons - Bart vs The Space...

Aşağıdaki oyunlar ise çalışan ancak problem çıkaranlar:

1. Fire and Ice : Ana karakter sprite'ı görünmüyor
2. Shadow of The Beast III : Ana karakter sprite'ı görünmüyor
3. Stardust : Ana karakter sprite'ı görünmüyor
4. Superfrog : Ana karakter sprite'ı görünüyor ancak eksik/hatalı
5. Project X SE : Ekranın sağ tarafında yaklaşık 1 cmlik bir çizme/silme işlemlerinin görüntülendiği bir alan var

Kaynak:
http://www.loriano.pwp.blueyonder.co.uk/review_of_the_minimig_v1.htm

# Commodore dünyası

1. Türk yapımı olan C64TPC (Connect C64 to PC) piyasaya

çıktı

2. High Voltage SID Collection'ın 48. güncellemesi yayınlandı

3. Oxyron bir DTV demosu yayınlayarak DTV scene'ine resmi olarak giriş yaptı

4. Breakpoint partisinde Commodore 64 demo yarışmasını 8 demo arasından Exceed, Resource ve The Dreams iş birliği olan "Cauldron" oyununun temasını kullanan aynı isimli demo kazandı.

5. X-2008 partisinin web sitesi açıldı. http://www.scs-trc.net/x2008

6. X-2008'de yayınlanacak demolar ile ilgili söylentiler şimdiden başladı. Kesin olmayan haberlere göre demo yayınlayacak gruplar: Digital Sounds System, Gheymaid Inc., Swappas with Attitude, Onslaught, Camelot (Cruzer demonun isminin "Meet the Camels" olacağını söylüyor. Bu Crest'in yıllardır çıkmak bilmeyen "Meet Crest" demosuna gönderme bir isim). Tüm bu haberler söylentiden ibaret ve belki de yalnızca espiri. Yine de X-2008'de yayınlanması muhtemel demolarla ilgili bize fikir veriyor.

7. Soldan sağa ya da sağdan sola kayabilen Shoot'em Up oyun editörü Sideways Seuck yayınlandı. http://www.seuckvault.co.uk

8. Breakpoint C64 Demo kategorisinde, Cauldron demosu ile birinciliği kazanan Resource & The Dreams & Exceed grupları, kazandıkları ödül parasını CSDB'ye (Commodore Scene Database http://noname.c64.org/csdb) bağışladılar.

9. Nintendo firması, Wii'nin Virtual Console'una Commodore 64 oyun klasiklerinin ekleneceğini açıkladı.

10. C64.sk sitesi her 1 Nisan'da yaptığı gibi bu sene de ziyaretçilerini espirili bir şekilde karşıladı, daha doğrusu karşılayamadı! Geçtiğimiz yıllarda 1 Nisan günleri genellikle c64.sk domainini atari.sk'ya yönlendiren site, bu sene zx spectrum haberlerinin yer aldığı www.raww.org sitesini tercih etti.

# 1541 Ultimate

## Nedir?

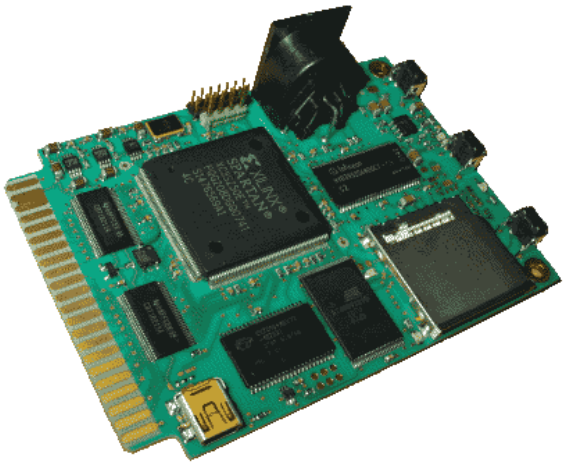
'1541 Ultimate' Commodore (C64, C128, C plus4, C16, vb.) için bir depolama çözümüdür. Bu donanım Commodore için 'gerçek' bir 1541 disk sürücüsü sağlar ve floppy disklerinizi modern bir SD-card yada MMC-card'ta saklama olanağı tanır. Gerçek bir sürücü ile bütünüyle uyumlu olması için için gerekli olan tüm donanımların birer replikasını içerir.

'1541 Ultimate' C64/C128 ile kartuş olarak kullanılabilindiği gibi serial bus'a sahip diğer Commodore modelleri ile de ayrı bir birim olarak kullanılabiliniyor. Kartuş modunda kullanıcının SD-card içeriğini görebilmesi için tek bir tuşa basması yeterli oluyor ve .d64 imajları disket sürücüye yükleniyor. Ayrı birim modunda ise kullanıcı, tuşları kullanarak klasörler arasında hareket edebilir ya da bus üzerindeki ikinci bir aygıta IEC komutları yollayabilir. Bu ikincil aygıt gerçek 1541 sürücüsü kadar uyumlu olmasa da FAT dosya sistemine direk olarak dosya yazma ve okuma sağlıyor.

## İçerik

'1541 Ultimate' kişisel bir hobi projesidir, Halihazırdaki Commodore kullanıcıları için oldukça kullanışlı olacak olgun bir cihaz haline gelmiştir. Kullanıcının gerçek 5.25 disketlerini SD-card'a kopyalamasını ve oradan çalıştırmasını sağlar ve bunu yaparken C64'ün gerçek bir donanım kullanılmadığını anlamamasını sağlar. Floppy diskler yaşlanmaya mahkumdur ve çoğu disketin okunamaz duruma gelmesine çok az kalmıştır. Yani Commodore meraklılarının yürüttüğü daha büyük bir "Koruma Projesi" kapsamında '1541 Ultimate' çok faydalı olabilir.

Bir firma olmamama rağmen '1541 Ultimate' donanımının Commodore meraklıları için ulaşılabilir kılmak istiyorum, çünkü bu aygıtın oldukça güçlü olduğuna ve sadece 1541 sürücüsü için bir yedek olmaktan daha fazlasını yapabilecek potansiyeli olduğuna inanıyorum. Umarım insanlar oldukça fazla ilgi ve istekle yapılmış olmasına rağmen bu kartın (board'un) kişisel bir proje olduğunu, resmi olarak satışa sunulacak bir ürün olmadığını anlayabilirler.



**Şekil 3. 1541 Ultimate**

## Ana Hatlar

1. Gerçek bir Commodore 1541 disk sürücüsünün tam emulasyonu.

2. FAT dosya sistemine direk erişim için İkincil 'IEC-sürücüsü'

3. FAT/FAT32 desteği, alt klasörler ve uzun dosya adları ile birlikte

4. C64 kartuş emulasyonu; Final Cartridge III, Action Replay,

Commodore 1750/1764 REU

5. D64 imajlarını listelemek, seçmek, yüklemek ve yaratmak için esnek dondurucu menüsü
6. Ayarlama Ekranı, yerel ayarlarla birlikte
7. 'Ayrı birim modu' - Mini-USB konnektörü ile enerji sağlanmakta, tuşlar ve IEC ile kontrol edilmekte

## Özellikler

1. Board boyutu: 68 x 96 mm
2. 250k-gate FPGA
3. Tam boy 6-pin DIN konnektör (IEC)
4. 512 kB SRAM
5. 2 MB Flash ROM
6. 32 MB SDRAM (sadece Plus versiyonu)
7. Tam boy MMC / SD card uyumlu (Mini/Micro SD adaptörle kullanılabilir)
8. Stereo ses çıkışı
9. Teyp port'una bağlantı için 6-pin expansion pin-header (C2N emulasyonu için), ya da gelecekteki bazı özellikler için.

Kaynak

www.1541ultimate.net

Çeviri

Ozan 'Ragnor' Emirhan Bayyurt

# Taşınabilir Aygıtlar

## Dokunmatik Ekranlı GP2X F-200

Dokunmatik ekranlı yeni GP2X karşınızda!

Taşınabilir multimedya konsollarına farklı bir bakış açısı getiren GP2X Oyun ve Multimedya Konsolu, dokunmatik ekranlı yeni modeli GP2X F-200 ile teknoloji meraklılarının ilgisine sunuldu. Açık Kaynaklı Linux işletim sistemine sahip GP2X F-200 yeni Firmware 4.0 ile birlikte geliyor. Yeni firmware ile GP2X F-200 32GB'a kadar SDHC kartları kullanabiliyor ve eklenen Multi-Tasking yetenekleriyle GP2X, fotoğraflarınıza bakarken yada e-kitap okurken müzik dinlemenize imkan sağlıyor. Geliştirilmiş video oynatıcısı ile DivX ve XviD biçiminde sıkıştırılmış filmleri herhangi bir değişiklik yapmadan altyazı desteği ile GP2X'inizde seyredebilirsiniz. GP2X'in kalbi olan MMSP2 işlemcisinin Video oynatmadaki yetenekleri sayesinde 720x480 çözünürlüğe kadar olan videoları bütünleşik ekranında otomatik olarak boyutlandırarak gösterirken, TV-Out seçeneğiyle de videoyu orijinal çözünürlüğüyle Televizyonunuza aktarmanızı sağlıyor. Tamamen yenilenmiş ses sistemi ile GP2X'in dahili hoparlörlerinden ve kulaklık çıkışından oldukça tatmin edici düzeyde ve kalitede ses alınabiliyor.


**Şekil 4. GP2X F-200**

Açık kaynaklı kodlu bir işletim sistemine sahip olmanın avantajı GP2X'e yazılan oyun ve programlara göz atıldığında daha iyi anlaşılıyor. Geliştirme kitleri internet üzerinden indirilebilen GP2X için her geçen gün yeni oyunlar ve uygulamalar yayınlanıyor. Bu uygulamalar arasında;

1. 50'nin üzerinde değişik bilgisayar sistemi için geliştirilmiş emülatörler, (Commodore 64/128/Amiga, Atari XT/ST/Lynx, Nintendo GameBoy/GameBoy Color/GameBoy Advance, Sega Master System/Genesis/Megadrive/Game Gear, M.A.M.E, CPS1, CPS2, Final Burn, Neo-Geo ve daha birçoğu…)
2. Klasik PC oyunlarını oynamanızı sağlayacak Interpreterler, (Duke3D, Quake, Doom, Rise of The Triad, Descent, ScummVM, Ultima, Transport Tycoon Deluxe ve birçoğu…)
3. GP2X'e özel olarak hazırlanmış oyun ve programlar ,
4. Diğer Linux versiyonlarından GP2X'e uyarlanmış birçok oyun ve programlar,

gibi bir çok uygulama yer alıyor.

**Şekil 5. GP2X Cradle**

GP2X Cradle, GP2X'in en önemli aksesuarlarından biri. Üzerinde bulunan 4 adet USB Host portu, TV ve Ses çıkışları, Paralel ve Seri portları bulunan GP2X Cradle, cihazın çok güçlü bir özelliği olan USB host özelliğini kullanımıza sunuyor. GP2X Cradle'a bağlayacağınız Klavye, Mouse, Game Pad, Joystick vb. USB aygıtlarla oyun zevkinizi arttırabilirsiniz. GP2X Cradle'a bağlayacağınız TV, Ses Sistemi ve 2 adet USB Game Pad ile odanızı bir arcade salonuna dönüştürmek ve eğlencenizi arkadaşlarınızla paylaşmanız da mümkün! Ayrıca Cradle'a bağlayacağınız USB Harddisk, Flash Bellek, Kart Okuyucu vb. depolama birimlerindeki film, müzik, oyun, resim ve e-kitaplarınızı da GP2X'inizde görüntüleyebilirsiniz. Gerekli sürücülere ve uygulamalara sahip olduğunuz sürece bütün USB cihazları GP2X'inizde kullanabilirsiniz. İsterseniz Webcam'inizi Cradle'a bağlayın ve resim çekin! (Cam2X)

Hemen belirtmekte fayda var ki GP2X'i kullanmak için herhangi bir şekilde Linux bilgisine veya bilgisayarınızda Linux işletim sistemi yüklü olmasına gerek yok. Ayrıca Windows XP/Vista işletim sistemleri de GP2X'inizi herhangi ek bir sürücüye ihtiyaç duymadan görüp ve kullanmaya başlayabiliyor.

www.gp2xtr.com

# Z80 İşlemcilere Yeni Bir Dil

Türker (Alcofribas) Gürevin

Z80 işlemci kullanan bilgisayarlar için C sözdizimi tabanlı yeni bir dil olan ccz80 ve buna ait IDE http://perso.orange.es/emilio.guerrerog adresinde duyuruldu. Komut satırında çalışan derleyici; ccz80 dilinde yazılmış bir koddan, Assembler'da binary çalışır dosya haline getirmek üzere bir ASM kodu üretiyor. Daha sonra bu ASM kodunu ister gerçek bir bilgisayarda isterseniz bir emülatörde kullanabilirsiniz.

Şu an için Standart kütüphane haricinde Spectrum, Amstrad CPC 464, CPC 6128 ve MSX kütüphaneleri mevcut. Bu derleyici ücretsizdir ancak kodu açık değildir.

Nasıl kullanılacağı sitesinde gayet detaylı anlatılıyor ve güzel çalışan bir IDE'si varsa da, biz Amstrad CPC'de test etmiş olduğum yöntemi adım adım görelim:

## Dosyalar

İlgili dosyaları ve fazlasını http://perso.orange.es/emilio.guerrerog veya http://www.ccz80.tk adresinden indirin.

Şu an gerekli olan dosyalar

1. ccz80 compiler
2. ccz80 standart kütüphanesi
3. Amstrad CPC 464 kütüphanesi
4. Amstrad CPC 6128 kütüphanesi

Diğerleri

1. ccz80 dili ve standart kütüphane dokümantasyonu
2. ccz80 dili IDE ve IDE dokümantasyonu
3. Amstrad CPC 6128 kütüphanesi dokümantasyonu
4. UltraEdit için CCZ80 sözdizimi dosyası
5. Spectrum kütüphanesi
6. Spectrum kütüphanesi dokümantasyonu
7. MSX kütüphanesi
8. MSX kütüphanesi dokümantasyonu
9. Pasmo programı eğer emülatörünüzde yoksa Cross Assembler olarak tavsiye ediliyor. http://www.arrakis.es/~ninsesabe/pasmo/#down

Bir klasör açın ve içine "ccz80.exe", "standard.ccz80" ve "Amstrad library cpc464.ccz80" dosyalarını zip klasörlerinden çıkararak kopyalayın.

## Programı Yazmak

Notepad'da yine aynı klasör içine test isminde(başka birşey de olabilir) bir txt dosyası oluşturun ve içine şu kodu yapıştırın:

```
// Use de Amstrad library
include "cpc464.ccz80";

// Declare a byte variable
// and initialize it with 32
byte i = 32;

// Loop of 224 pass
repeat (224)

// Call to printc function
// (included in cpc464.ccz80 library)
// and post-increment the i variable
printc(i++);

// Ends program an return to BASIC
return;
```

## Kodu Derlemek

Windows'da Başlat/Çalıştır'dan (Start/Run) cmd komutunu vererek komut satırına geçin. Cd komutunu ve doğru yolu vererek oluşturduğunuz klasörün içine girin.

Klasörün içinde iseniz şu satırı yazın:

```
ccz80 program.ccz80 /org=#A000
```

Herhangi bir hata yoksa şu anda çalışma klasörünüz içinde test.asm isimli bir dosya oluşmuş olması lazım.

WinApe emülatörünü çalıştırın ve F3'e basarak Assembler ekranına geçin. File/Open ile test.asm dosyasını yükleyin ve Ctrl+F9 ile kodunuzu Assemble edin.

Emülatör ekranına dönün ve CALL &A000 komutunu verin. Sürpriz olmasa bile sonucu seyredin ;)

Bana ulaşmak isterseniz:

8bitmicro (at) gmail (nokta) com

# Pandora'nın Kutusu Açılıyor

Emir (Skate) Akaydın

2007 Ekim'inde saygın teknoloji portallarından olan engadget.com bir grup geliştiricinin gizemli bir cihaz üzerinde çalıştığını izleyenlerine duyurdu. Pandora adıyla duyurulan bu cihazın özellikleri avucunun içinde bir canavar taşımak isteyen bütün "geek"lerin kalbinin heyacanla çarpmasına yetti. Bahsedilen cihaz İngiltere, Almanya ve Türkiye'deki GP2X distribütörlerinin ortak çalışmaları sonucu hergeçen gün hayalden gerçeğe doğru yolculuğunda hızla devam ediyor. İlk zamanlarda "vaporware" olarak nitelendirilen cihazın ana PCB resimlerinin kısa süre önce yayınlanması ve Texas Instruments'ın Dallas'taki merkezinde düzenlenen geliştiriciler konferansında tanıtılmasıyla ne kadar gerçek olduğu ortaya çıkmış oldu.

Peki Pandora ne vaat ediyordu? Asus, HP, Acer gibi bilgisayar devlerinin ultra ucuz bilgisayar üzerinde çalıştığı bu günlerde Pandora projesini ortaya koymak mantıklı mıydı?

Geliştiricilere göre Pandora çok farklı bir cihaz. Pandora'nın alternatifi olarak gözükebilecek minik laptoplarda uygun oyun kontrolleri olmadığı gibi oyun oynamaya yönelik ergonomik bir tasarım yok. Üstelik pil ömürleri yetersiz ve 3D donanımları yok ya da kısıtlanmış. Fakat pandora bir oyun konsolu olma özelliğinin yanı sıra çeşitli ofis ve internet uygulamalarını çalıştırabilme potansiyeli ile diğer ultra mobil PC'lerin pazar payına ortak olabilir.

**Şekil 1. Pandora**

Pandora'nın geliştiricileri cihazı "Gerçek oyun kontrollerine sahip ultra mobil kişisel bilgisayar" olarak tanımlıyor. Cihazın ilk resimlerini ve teknik özelliklerine göz attığımızda gerçekten taşınabilir bir oyun konsolundan bulunması gereken bütün kontrollere sahip olduğunu görüyoruz. Bunlar:

1. 2 adet Analog Kontrol
2. 1 adet Dijital Yön Tuş Takımı (D-Pad)
3. 2 adet Tetik Tuşu (R ve L)
4. 4 adet Aksiyon Tuşu
5. Star, Select ve Menu Düğmeleri

Bunun yanında 800x480 çözünürlüğünde dokunmatik ekran, Q Klavye, USB Host portu, Kablosuz Ağ, Çift SDHC slotu gibi özellikleri cihazın aynı zamanda kişisel bilgisayar özelliğini ortaya koyuyor.

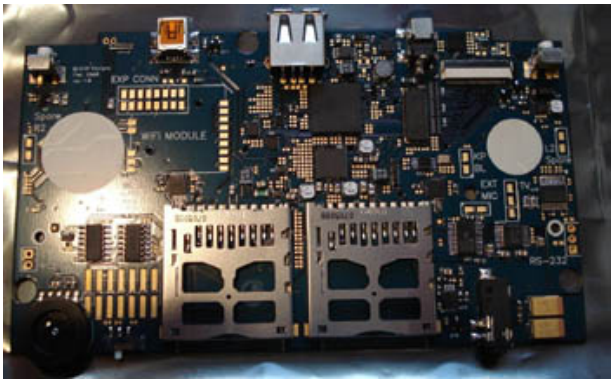
Bu kadar geniş bir kullanım alanı sunmayı vaat eden Pandora'nın bunu destekleyen bir donanımı var.

1. Texas Instruments OMAP 3530 SoC
2. ARM® Cortex™-A8 CPU
3. PowerVR SGX GPU (OpenGL 2.0 ES uyumlu)
4. 800x480 4.3 inch 16.7 milyon renkli dokunmatik LCD
5. 256 MB Dahili Flash Bellek
6. 128 MB 333 Mhz DDR Ram
7. Wifi 802.11b/g Kablosuz Ağ
8. Çift SDHC Kart Slotları
9. Dual Analog ve Dijital oyun kontrolleri
10. 43 tuşlu QWERTY ve numerik klavye
11. TV Çıkışı
12. Yüksek Hızlı USB Host

Pandora bu donanımı tamamen açık kaynaklı bir Linux işletim sistemiyle kontrol ediyor. Açık kaynak kodlu bir işletim sistemine sahip olmanın en büyük avantajı hızla geliştirilecek olan çeşitli uygulamalar, oyunlar ve demolar. GP2X topluluğu ise efsanevi C64, MAME, NES, SNES, GBA, N64, PSX ve AMIGA gibi sistemlerin emülatörlerinin cihazın çıkışıyla birlikte hazır olacağını bekliyor. Bunun sebebi ise cihazın anakartlarının çeşitli geliştiricilere ulaştırılacağı (veya ulaştırıldığı) haberlerinin meydana çıkması. Bu ekstra motivasyon Pandorayı bekleyen kullanıcılarını daha sabırsız hale getiriyor.

Geliştiricilerin tanımına göre Pandora kullanımı çok basit menülere sahip olacak ve ilk defa kullananların bile yabancılık çekmeyeceği şekilde düzenlenecek. Ayrıca internet tarayıcısı, anında mesajlaşma ve çeşitli temel uygulamaların firmware içinde yer alacağı belirtiliyor. Bunun dışında ogg, mp3, divx, xvid gibi temel medya biçimlerini oynatabilen uygulamalar sisteme dahil edilmiş.

Pandora'nın boyutları hemen hemen bir Nintendo DS kadar. Bunun yanında performans olarak günümüzdeki bütün taşınabilir oyun konsollarından daha güçlü. Peki bu performansı ne kadar sürdürebiliyor? Lithium-Ion pili sayesinde Pandora'nın normal kullanımda pili 6 saat dayanıyor. Fakat OMAP 3530'un özellikleri arasında işlemci gücünü uygulamaya göre otomatik olarak ayarlayan bir sistem var. Kısacası çok güç gerektirmeyen uygulamalarda işlemci 10 Mhz'e kadar düşebiliyor. Bu da cihazın bir mp3 çalma işlemini yerine getirmek için yeterli bir hız.




**Şekil 2. Pandora anakarti**

Pandora scenerlar açısından da oldukça ilginç bir platform. Hem sabit bir donanıma sahip, hem de gelişmiş özelliklere. Bu açıdan, yıllardır Commodore 64 gibi 8-bit platformlarda ünlenmiş "aynı şartlarda kapışma", Pandora ile daha geniş ufuklarda bizleri bekliyor.

Piyasaya sürüleceği tarih tam olarak kesinlik kazanmamış olsa da Mayıs/Haziran aylarında Pandora'nun raflardaki yerini alacağı tahmin ediliyor.

Emir 'Skate' Akaydın

# Demo İnceleme: Cauldron

Emir (Skate) Akaydın

Şemseddin (Endo) Moldibi

Kürşad (Hydrogen) Karamahmutoğlu

## Giriş:

Breakpoint'te C64 demo yarışmasını kazanan Cauldron isimli demo, tema olarak C64'ün aynı isimli klasik oyunundan esinlenilerek hazırlanmış. Demo'nün büyük bir bölümü Murphy / Exceed / Ex-Resource tarafından 1997 yılında kodlanmış, bugüne dek yayınlanmamış efektlerden oluşmakta.

Demo, Cauldron oyununun giriş ekranındaki grafik ve animasyonlarıyla başlıyor. İlk olarak sağa doğru süpürgeyle uçarak ilerleyin ve turkuvaz anahtarı alın, ekranın yukarısındaki magic puanına dikkat edin, o puan bittiğinde ölürsünüz. Magic puanınızı doldurmak için etrafta görebileceğiniz kıvılcımlara dokunabilirsiniz. Bunun için yere inmeniz ve yürüyerek ilerlemeniz gerekir, yere inmek için boş alan bulmalısınız.. Demonun esas amacı balkabaklarından kaçarak… Ee. galiba oyuna kaptırdık biz! Tamam, baştan alıyoruz.

Demo, Cauldron oyununun giriş ekranındaki grafik ve animasyonlarıyla başlıyor. Ekranın yukarısında multicolor bir Cauldron logosu beliriyor, logonun içerisinde color-cycle efekti ile plazma benzeri bir görüntü oluşturulmuş.

**Şekil 1.**

Endo: Giriş olarak çok hoş düşünülmüş bence.

Skate: Oyunun bir fanı olarak demonun orijinal oyun grafikleri ile başlayan girişi bana oldukça nostaljik duygular yaşattı. Zannedersem hedef kitle olarak benim gibileri seçmişler.

Hydrogen: Logonun renkleri güzel seçilmiş. Efekt de çok hoş uygulanmış.

Skate: Sanırım logo orijinal oyundan alınmamış ancak oyundaki forma bağlı kalınmış. Cycle efektini ise çok güzel tasarlamışlar.

Demo bir 4x4 efekt ile devam ediyor. Efektin üzerinde ağaç silueti grafiği var.

**Şekil 2.**

Skate: Bu efekt her ne kadar plazmatik görünse de normal bir plazmadan farklı olduğu ilk bakışta anlaşılıyor. Efekt alanında iki tane magnetik merkez göze çarpıyor. Demonun coderlarından Oswald'ın CSDB'de dediğine göre, bu ve bundan sonraki iki 4x4 efekt, iki texture distortion tablosunun toplanmasıyla oluşturulan harita üzerine texture basılarak elde edilmiş. Bu durumda her ne kadar düşük çözünürlüklü bir efekt olsa da sonuç hız ve görüntü olarak tatmin edici.

Hydrogen: Bu tarz 4x4 efektlere günümüz scene'inde çok rağbet edilmiyor. Bu trend en çok 90'ların ikinci yarısında popülerdi. Bu sebeple bu efektlerin 97 yılında yapıldığını belirtmek gereği duymuşlar.

Endo: Aynı şeyi söyleyecektim ben de. 97 yılında yapılmış olduğu düşüldüğünde güzel bir efekt. :)

Hydrogen: Ancak 2008'de yayınlanmış bir demo için… (burada Hydrogen birkaç kez farklı şekillerde cümlenin devamını getirmeyi dener) ya salla ya saçmaladım resmen.

Skate: 64 Times günlerine dönüş :)

Hydrogen: Bu efektlerin Coma ve Resource gruplarının 97'de yayınladığı Void isimli co-op demoda neden yer almadığını merak ettim.

Skate: Evet, Void'de bir sürü 4x4 efekt vardı.

Hydrogen: Tamamını o zamanlar Coma'da olan Oswald kodlamıştı.

Endo: 97 yılında yapılmış olduğu düşüldüğünde güzel bir efekt. :)

Cauldron oyunun kapak grafiği olan cadı resmi ekrana geliyor. Aynı zamanda cadının önündeki kazandan baloncuklar çıkıyor.

**Şekil 3.**

Hydrogen: Leon bu tarz, oranları çok belirgin olmayan resimleri pixellemekte daha başarılı.

Endo: Bence baloncuklar daha kaliteli olabilirmiş.

Hydrogen: Parazit yapma Endo. Renkler oldukça iyi seçilmiş. Detayları küçük alanlarda çizmek oldukça zordur. Bu resimde cadının yüzü, kolları ve saçları oldukça küçük alanda başarılı bir şekilde piksellenmiş.

Endo: Cadının saçlarındaki renkler çok güzel seçilmiş.

Hydrogen: Kazandaki sıvıyı bir iki sprite animasyonuyla hareketlendirselermiş hoş olurmuş.

Skate: Kazandaki sıvı bana domates çorbasını hatırlattı. Bir coder'ın bir grafiğe yorumu bu kadar oluyor :)

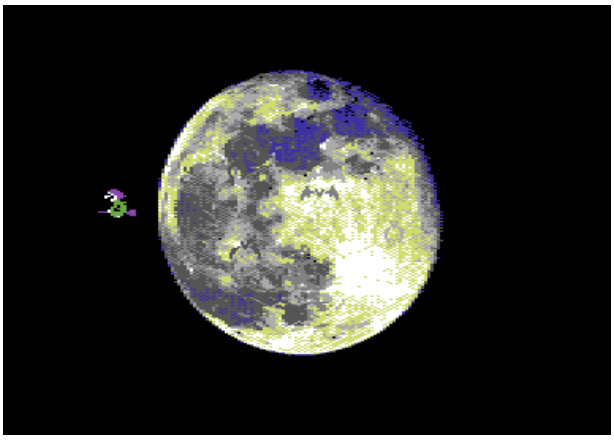
**Şekil 4.**

Yine bir 4x4 efekti farklı bir ağaç silueti ile ekrana geliyor.

Skate: Bu efekt öncekinden farklı olarak biraz daha küresel bir görüntü veriyor. Bir tür polar koordinat efektini andırıyor.

Endo: Küresel görünüm gayet hoş olmuş.

Ekrana bir ay resmi geliyor ve önünden yarasalar ve süpürge ile uçan cadı geçiyor.



**Şekil 5.**

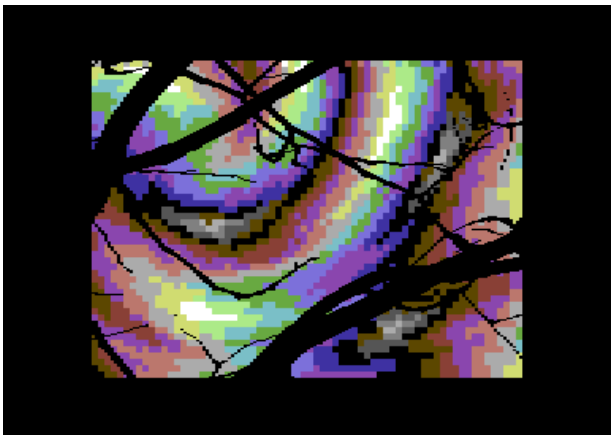
Skate: Özellikle ay grafiği çok kaliteli olmuş ancak kullanılan mavi renk sanki ay üzerinde su varmış gibi bir his uyandırmış. Cadı konseptine gayet uygun olmuş. Ben olsam hidden-part olarak E.T. de koyardım. :)

Endo: Bence yarasalar siyah olsaymış daha güzel olurmuş.

Skate: Keşke bir işe yarasalar :)

Hydrogen: Arkadaşlar denilecek her şeyi dediler. Bu tarz efektleri kesinlikle evinizde denemeyin. :)

Bir diğer 4x4 efekti ekrana geliyor.



**Şekil 6.**

Hydrogen: Bence en başarılı görsele sahip 4x4 efekt bu olmuş. Efektin belli bir merkezi olmaması dinamizmini arttırıyor.

Skate: Bence de bu efekt üç 4x4 efekt içinde en başarılısı. Bunda net bir şekilde texture'ların bir path üzerinden kaydığını ve path'in dinamik olarak değiştiğini gözlemleyebiliyoruz.

Elinde balkabağı tutan cadı grafiği ekrana geliyor. Cadının arkasında 4x4 bir ray efekti beliriyor.

**Şekil 7.**

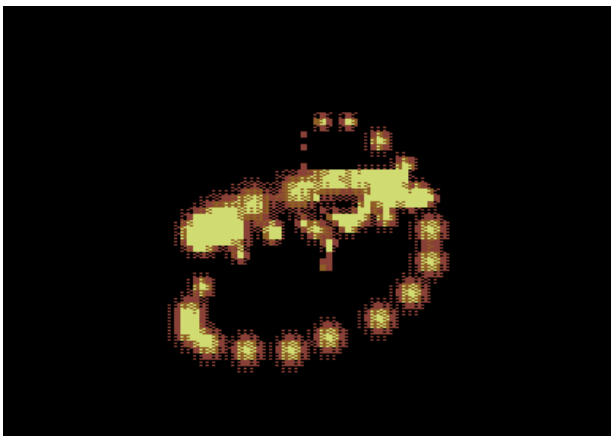
Hydrogen: Leon'un en başarılı resimlerinden biri. Yüksek kontrastlı piksellemenin başarıyla kullanılmış olması resmin 16 renkten daha zengin bir palete sahip olduğu izlenimini uyandırıyor. Ancak efektin resim ile yeterince uyumlu birleştirilmediğini düşünüyorum. 4x4 güzel sonuç vermemiş.

Endo: Grafik çok kaliteli, özellikle kıyafetin dokusu çok güzel.

Skate: Bu grafik önceki cadı grafiğiyle yüksek oranda uyumluluk gösteriyor. Aynı zamanda demonun akışındaki temanın devamlılığını perçinliyor. Grafiğin kalitesi hakkında Hydrogen ve Endo'nun söylediklerine katılıyorum. Efektin diğer 4x4 efektlerden farklı olarak 2008'de kodlanmış olması, biraz kolaya kaçıldığı hissi uyandırıyor. Daha kaliteli ve yüksek çözünürlüklü bir ray efekti yapılabilirmiş.

Grafiğin ardından ekrana bir bob efekti geliyor.

Skate: Boblar bana particle hissi verdi. Bir tür shade bob kullanılmış. Bobların büyüyüp küçülmesi efekte 3 boyutlu bir his kazandırmış. Ancak bence boblar yine 4x4lük alanlarda kayacağına, wu-pixel tarzı bir yöntemle daha yumuşak koordinat geçişleri sağlanmış olsaydı efektin yaratacağı etki çok daha yüksek olurdu.



**Şekil 8.**

Endo: Efekt gayet güzel görünmesine rağmen çözünürlüğü daha yüksek olsaymış daha iyi olurmuş.

Hydrogen: Hayret Endo sinüslere dair bir şey demedi.

Endo: A evet lan. Sinüslerden ziyade 3 boyutlu hissi uyandırması gayet hoş olmuş.

Bob efektinin ardından ekrana bir balkabağı grafiği geliyor.

**Şekil 9.**

Endo: Demodaki diğer grafiklere nazaran bu grafik biraz renksiz görünüyor.

Hydrogen: Kabağın ortasındaki koyu gölge şeridi geometrinin bozuk görünmesine yol açıyor. Bu da 3 boyut hissine darbe vuruyor.

Endo: Evet, grafik 3 boyutlu çizilmiş olmasına rağmen 3 boyutlu değilmiş gibi görünüyor.

Skate: Alakasız bir biçimde benim gözüme çarpan olay, bu grafiğin ay grafiği önünden geçen yarasalar ve cadı bölümündeki ayın büyüklüğüne oldukça yakın dairesel bir alana çizilmiş olması. Bu yüzden bu grafiği gördüğümde demonun geneliyle bir bütünlük hissi yarattı benim üzerimde. Renk sayısının düşük olması ise göze batıyor.

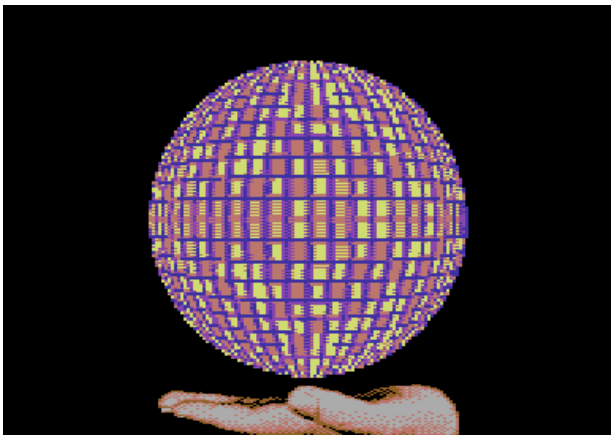
Hydrogen: Demek demolarda 3-5 top çizsek koherasyon olacak.

Skate: Örneğin Cycle demosunun girişindeki dairesel grafik de o demonun kimliği haline gelmiştir. Aslında evet, dairesel 3-5 grafik işi götürüyor. :)

Hydrogen: Hohoho.

Skate: Ben ciddiydim. :(

Balkabağı grafiğinin bulunduğu alan bir fade efektiyle el üzerinde duran disko topuna dönüşüyor.



**Şekil 10.**

Hydrogen: Ah, koherasyon çarpması yaşadım. :)

Skate: Oswald'ın dediğine göre bu part Boogie Factor / Fairlight demosuna bir tür gönderme ya da tribute. Ancak benim görüşüm disko topu gibi demonun konseptine uymayan bir obje yerine büyücü küresi tarzı bir efekt yapılması daha uygun olurmuş. Tahmin ediyorum aynı efekt mantığını kullanarak, konsepte daha uygun bir sonuç elde etmek çok zor olmazdı.

Hydrogen: Bence Oswald gerçekte suçsuz olabilir. Belki o sadece bir piyondu. :)

Endo: Bence de genel konsepte pek uygun düşmemiş.

Yukarıya doğru kayan yazı ve arkasından sağdan sola doğru geçen yarasalar ile demo sona eriyor.


**Şekil 11.**

Skate: Bir bitiş bölümü olarak sıradan bir bölüm olmuş. Sıradanlığın yanı sıra demo geneline göre de biraz fazla basit kaçmış.

# Demo ile ilgili genel yorumlar:

Endo: Müziğin ritmi, yer yer yavaşlamalar olsa da demonun geneline göre biraz hızlı kaçmış.

Skate: Aslında müziğin hızıyla ilgili asıl problem, daha yumuşak bir his bırakan efektlerde gereğinden fazla hızlı iken, hızlı bir müziği kaldırabilecek bölümlerde daha yavaş ritimlerin kullanılmış olması. Kısacası efekt-müzik uyumu üzerinde fazla kafa yorulmamış. Zaten demo genelinde de müziğin ritmini parametre alan bir efekt mevcut değil. Bunların yanı sıra müzik tek başına değerlendirildiğinde genel olarak hızlı ritimlerin üzerine mistik efektlerden oluşuyor. Zaman zaman ritim hızı efektleri kaldıramayacak bir noktaya gelebiliyor.

Hydrogen: Bence ritmin normal enstrümanlarla çalınamayacak kadar hızlı olması müziği fazla deneysel yapmış. Efekt-müzik uyumu genel olarak birçok demoda çok fazla önemsenmiyor. Benim daha çok dikkat ettiğim şey rahatsız edici bir uyumsuzluk olmaması. Bu demo bence sınırda. Gene de müziğin demoya olumlu etki ettiğini söyleyebilirim.

Endo: Sonuç olarak efektlerde bir yenilik olmaması ve kısa bir demo olmasına karşın, konseptinden dolayı iyi bir izlenim bırakıyor.

# Demo yapımcıları:

*Kod*

Edhellon / Resource

Murphy / Exceed / Ex-Resource

Ninja / The Dreams

Oswald / Resource

*Müzik*

Linus / Resource / Viruz

*Grafik*

AmN / Resource

Leon / Chorus / Singular

*Dizayn*

Oswald / Resource

*Loader*

Doc Bacardi / The Dreams

*Yardım*

Clarence / Chorus

# Değerlendirme:

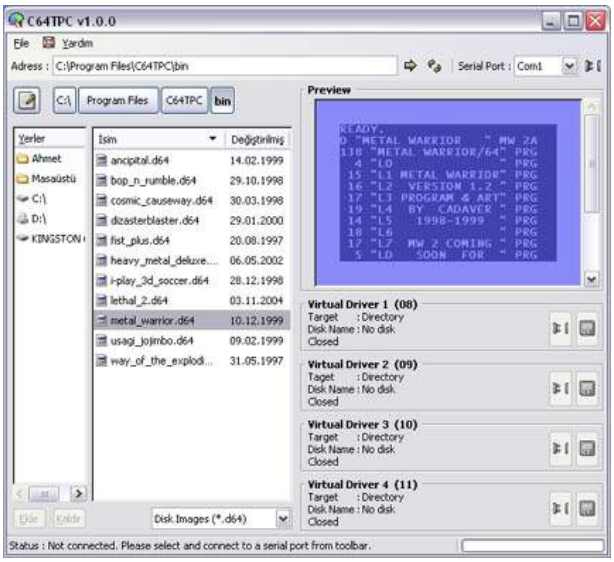
Endo: %76 Hydrogen: %68 Skate: %78

Ortalama: %74

# C64TPC - Proje Günlüğü

Ahmet Zeki Eymür

Bir süredir aklımda, TPC'nin geliştirilmesi sırasında yaşadıklarımı aradan fazla zaman geçmeden taze taze yazıya dökmek vardı. Plazma için TPC hakkında bir yazı yazmam istendiğinde kendime güzel bir ortam bulmuş oldum. TPC hem yazılım, hem donanım, hem de firmware ayakları olan bir proje. Bu yazıda okuyucuların proje hakkında teknik ayrıntılar bulabilmelerini sağlamak istiyorum. İlk önce kısaca C64TPC'nin ne olduğunu açıklayım.




**Şekil 1.**

## C64TPC

C64TPC, C64'ü ve PC'yi seri portlarından birbirine bağlayarak C64'ün PC'nin veri depolama aygıtlarına erişimini mümkün kılan bir yazılım ve donanım birimidir. PC'nin sabit diskinde, CD/DVD sürücüsünde vb. olan bir dosyayı doğrudan C64'e yükleminizi ya da tam tersi C64'ün belleğinde bulunan bir dosyayı PC'de istediğiniz bir konuma kaydetmenizi sağlar. Bunu yaparken C64 tarafında ilave bir yazılıma ihtiyaç duymazsınız. Daha önce alıştığınız şekilde LOAD, SAVE, SCRATCH vb. komutları kullanırsınız. Yazılım C64'ten gelen komutlara göre PC tarafında gereken işlemleri yapar. Donanım ise farklı mimarideki bu iki bilgisayarın haberleşebilmesi için gereken tüm alt seviyeli işlemleri halleder.

**Şekil 2.**

## GELİŞTİRME SÜRECİ

C64TPC, bu güne kadar üzerinde çalıştığım en karmaşık projeydi ve benim için çok öğretici oldu. Gerçi insana en son neyle ilgilendiyse en zoru oymuş gibi geliyor olabilir. Çünkü karşılaştığı zorlukların ve sıkıntıların anıları daha taze oluyor. Geliştirme süreci boyunca en az iki kere TPC'yi pencereden aşağı atmak istedim. :) İki kere de vazgeçmeyi ciddi olarak düşündüm. Bu tür uzun soluklu işlerde vazgeçmek de can yakıcı. Çünkü önceki emekleriniz ve zamanınızın çöp olmasına razı olmak kolay değil. Şimdi hatırladım, bir kere de üzerine hızlıca basıp ezmek istemiştim:).

Bu kadar sıkıntı yaşamamın büyük nedeni osiloskop ve lojik analizer gibi profesyonel ölçüm cihazlarına sahip olmamamdı. Bu cihazları birer cümleyle açıklamam yerinde olur sanırım. Osiloskop elektrik sinyallerindeki zamana göre değişimin grafiğini çizer. Lojik analizer ise dijital hatlardaki zamana göre lojik (0/1) değişimi gösterir.

Benim iki en büyük yardımcım "Inside Commodore DOS" ve "C64 Programcının El Kitabı" oldu. Zaten projeye başlamak için gerekli cesareti toplayabilmemi Programcının El Kitabında IEC portundaki hatların haberleşme sırasındaki lojik diagramlarının verilmiş olması sağladı. Programcının el kitabında bu tür bir bilginin bulunması bana garip gelmişti ama o diagramlar olmasaydı bu proje lojik analizersiz yapılamazdı. Elimde değişik durumları kapsayan üç diagram olduğu halde doldurmam gereken boşluklar hala vardı ve onları bulmak için benim elimde deneme yanılma yönteminden başka bir şey yoktu.

İşte en sıkıcı ve zor anlar onlar oldu. IEC portundaki hatların diagramlarda göremediğim lojik durumları için tahminler geliştirip defalarca deneme yapmak durumunda kaldım. Bir ara pes edip devreyi 8KB RAM takviyesiyle sadece hatların durumlarını kaydedip PC'ye gönderen basit bir analizere çevirmeyi bile düşündüm. Ancak zaman kaybetmek istemiyordum ve buna gerek kalmadan istediğim bilgileri deneme yanılma yoluyla derlemeyi başardım. Ancak bunu bir daha denemem :)

**Şekil 3.**

# IEC PROTOKOLU

Bu gün aklımıza seri port deyince RS232 geliyor. PC'ler seri iletişim için bu standardı kullanıyor. Ancak C64'ün seri portu başka bir standardı takip eder: IEC. TPC'nin donanımı, IEC ve RS232 arasında çift yönlü protokol uygunlaştırıcı olarak çalışır. Muhtemelen siz de benim gibi IEC'nin Commodore tarafından geliştirilmiş bir protokol olduğunu düşünmüşsünüzdür. Çünkü başka bir IEC aygıt duymadık. Ancak araştırmaya başlayınca gördüm ki öyle değilmiş.

Bu protokolün ardından yine bilindik başka bir şirket çıktı: Hewlett-Packard.Bu protokolü HP 1960'ların sonlarında geliştirmiş ve adını HP-IB (Hewlett-Packard Instrument Bus) koymuş. HP bu protokolü geliştirdiği multimetre ve lojik analizer gibi test ve ölçüm cihazlarının kendi aralarında ya da bilgisayarlar ile iletişimi için kullanıyormuş. Diğer üreticiler de bu protokolü kullanmışlar ama başka bir isimle: GPIB (General Purpose Interface Bus). Protokol, 1975 yılında benim doğumumla beraber IEEE (Institute of Electrical and Electronics Engineers) tarafından standartlaştırılmış :) IEEE bu standarta "IEEE-488" ismini koymuş. Peki biz buna neden IEC diyoruz. Çünkü Commodore'un dokümanlarında hep IEC adı geçiyor. HP-IB, aralarında ANSI (American National Standards Institute) ve IEC'nin de (International Electrotechnical Commission)olduğu bir kaç değişik komisyon tarafından da değişik isimler altında standartlaştırılmış. Sanırım Commodore firması bilgisayarlarının tasarımında IEC tarafından yayınlanan belgeleri baz almış.

İşte C64'ün seri portu bu protokolün daha az sinyalle ve daha düşük hızda çalışan bir gerçeklemesidir. Aklınıza, "İhtiyaç duyulmayan sinyallerin kullanılmaması tamam da, neden daha yavaş?" sorusu gelmiş olabilir. Önce şaşırtıcı cevabı vereyim sonra açıklayım. Nedeni, VIC çipi. 1541 disket sürücünüzü kullanırken LOAD ve SAVE işlemleri arasında anlamlı bir süre farkı yoktur. Ancak TPC'yi kullanırken aynı dosyayı yüklediğinizden (LOAD) çok daha hızlı kaydedebildiğinizi görürsünüz. Bunun nedeni TPC'nin mekanik parça içermemesi ve işlemin gerçekleşme hızının sadece veri transferin hızıyla ilişkili olmasıdır.

Yani bir program ya da dosya belleğe yüklenirken veri transferi daha yavaş, diske kaydedilirken daha hızlı olmaktadır. Bilgi akışı C64'ten dış dünyaya doğru iken (SAVE) veri bitleri arasında IEC standardı gereği 20us beklenir. Bilgi akışı dış dünyadan C64'e doğruyken ise (LOAD) veri bitleri arasında, IEC standartı 20us beklemeyi gerektirdiği halde 60us beklenir. Çünkü VIC çipi C64'ün mikro işlemcisinde 40us'ye varan kesmeler oluşturabilmektedir. Eğer böyle yapılmasaydı iletişim sırasında oluşabilecek bir VIC kesmesi 1 hatta 2 bitin C64 tarafından algılanamamasına yol açabilirdi. Bu yüzden her bitin transferi sırasında en kötü durumun oluştuğu yani bitin gönderilmeye başladığı anda bir VIC kesmesi oluştuğu farz edilerek (40us(VIC kesmesi max.) + 20us(IEC standartı gereği min.)) = 60us beklenir.

# EMULASYON

Zamanında Commodore firmasının disket sürücülere yaklaşımı çok farklı olmuş. 1541 disket sürücüsünün içinde bir C64 daha var desem sanırım abartmış olmam. 1541 disket sürücüsünün içinde C64'te kullanılan 6510 mikro işlemcisiyle arasında neredeyse hiç bir fark bulunmayan 6502 işlemcisi bulunur. Bu işlemciler o kadar aynıdır ki Frodo emülatörünün kaynak kodunu incelediğimde iki işlemci için aynı kaynak kod dosyasının kullanıldığını gördüm. Bu işlemci her 10ms de bir disket sürücü kontrolorü (FDC) gibi çalışır. Kalan zamanda ise Commodore DOS'u işletir. Evet MS-DOS gibi. MS-DOS zamanında koca işletim sitemine neden sadece Disk İşletim Sistemi adı verilmiş diye merak ederdim. Commodore bilgisayarları açısından isimde yadırganacak bir şey yok. Çünkü bu sistem 1541 ROM'unda gömülü olarak bulunuyor ve Disk sürücünün işlemcisi tarafından işletiliyor. Yani kelimelerin hakkını vere vere adını hakediyor: "Commodore Disk İşletim Sistemi".

Disk sürücü kendi işlemcisine ve o zamanlar için önemli bir rakam olan 2KB RAM'e sahip olunca sıra dışı kullanım şekilleri de başlamış. Oyun ve demo yazarları hesaplama amacıyla disk sürücünün işlemcisini, daha fazla hafıza için RAM'ini kullanmaya başlamışlar. Bir önemli sıra dışı kullanım şeklide daha hızlı veri transferi için kendi protokollerini geliştirmeleri olmuş. Yani 1541 disket sürücüsü dediğimizde aslında SID ve VIC çipleri olmayan bir C64'ten bahsediyoruz sayılır.

Peki TPC ilerde 1541 emulasyonu yapabilir mi? Bu konu üzerinde düşünmeye başladım. VICE'ın ve Frodo'nun kodlarını inceledim. 1541 ROM'unu da inceledim. İlk önce IEC transferini hangi entegrenin yönettiğini anlamaya çalıştım. Çok bacaklı VIA çipinin transferin ayrıntılarını kotardığını ve merkezi mikro işlemcinin verileri ondan bayt-bayt aldığını umuyordum. Çünkü PC'nin gerçek bir IEC portu yoktu ve PC, TPC'nin RS232 tarafındaydı. Ancak maalesef umduğum gibi olmadığını gördüm. IEC transferinin her anı bit-bit 6502 işlemcisi tarafından kontrol ediliyordu. Bu durumda ya yazılım ile bir IEC portu simülasyonu yapacaktım ya da 1541 ROM'unda bulunan IEC rutinlerini yamalayacaktım. Daha önce bahsettiğim "Inside Commodore DOS" kitabında 1541 ROM'unun ayrıntılı bir analizi bulunduğu için ilgili rutinlerin başlangıç adreslerini kolayca bulabiliyordum. Bu adreslerdeki komutu 6502 işlemcisinin kullanılmayan bir opcode'u ile değiştirebilir sonra Frodo'dan aldığım 6502 emulasyonu yapan kaynak kodları değiştirerek o opcode'a istediğim görevleri yükleyebilirdim.

Ancak daha sonra IEC portunun benim hiç hayal etmediğim kadar sıra dışı şekillerde kullanıldığını öğrendim. Foruma yazdığım soruya cevap veren Nightlord ATN hattını DATA hattı gibi kullanan loaderların olduğundan bahsediyordu. Başarılı bir emulasyon için IEC hatlarının durumlarının saniyede bir-kaç yüz binkere PC'ye transfer edilmesi şart görünüyordu. Bunun için gereken transfer hızına ise seri port ile çıkmak mümkün değil. Yani emulasyon konusundan önce TPC'nin USB versiyonunun yapılması gerekiyor. Ondan sonra VICE ya da Frodo gibi açık kaynak kodlu bir projeden alınacak kodlarla TPC yazılımına 1541 emulasyonu rahatlıkla yaptırılabilir.

Ahmet Zeki EYMÜR

# NatAmi'ye İlk Bakış

Gökhan (LW3D) Sönmez

## Natami, Amiga severlerin yıllardır beklediği ürün mü? Yeni bir Hayal mi?

Natami, Amiga severlerin yıllardır beklediği ürün mü? Yeni bir Hayal mi?

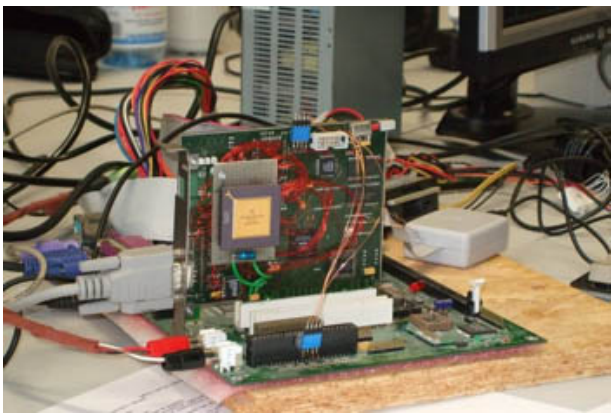
Commodore firmasının 1994 yılında iflas etmesinin ardından, dönemin en başarılı bilgisayarlarından biri olan Amiga'nın geleceği konusunda birçok öneri ve fikir ortaya atıldı. Bu fikirlerin hemen hemen hiçbiri hayata geçmezken, kaybedilen yıllar, Commodore firmasının son yıllarında zaten yeterince geliştirilemeyen Amiga'nın rakiplerine karşı daha da zayıf duruma düşmesine sebep oldu. Amiga'yı eski klasik yapısıyla, rakiplerine karşı üstünlük sağlayacak şekilde yeniden üretmek ve başarı sağlamak bir noktadan sonra mümkün olmadı. Bunun üzerine Amiga'nın üstün yönlerinden biri olan işletim sistemini, günümüz bileşenlerinden oluşan modern bir anakarta uyarlamak çok makul ve mantıklı göründü. Benzer yaklaşım, Amiga gibi 68000 serisi işlemci kullanan Macintosh bilgisayarları içinde bir çözüm olarak görülmüştü. Ama ne yazık ki, bugün için bu girişimin pek başarılı olduğunu söylemek mümkün değil. AmigaOne olarak adlandırılan ve PowerPC işlemcisini alt yapı olarak kullanan ve modern PC bileşenlerini destekleyen sistem artık üretilmiyor. İşletim sisteminin Power Macintosh'lara uyarlandığı, hatta Sony Playstation 3'e uyarlanması için çalışmaların sürdüğü iddialarının ileri sürüldüğü günümüzde, Amiga'nın yeni bir PowerPC board ya da tasarımla hayatına devam etmesi çok zor görünüyor. Peki sevdiğimiz, Amiga artık yok mu olacak? Bir daha hiç üretilmeyecek mi?

Cevap vermek zor...Ama Minimig, Clone-A ve Natami gibi projeler, klasik Amiga severler için birer ümit olarak karşımızda.

## Amiga Çipleri Yeniden hayat Buluyor

Amiga'yı Amiga yapan işletim sistemi miydi, özel çipleri miydi karar vermek zor. Kesin olan birşey var ki, özel işlemciler üretmek geçtiğimiz on yılda çok kolay değildi. Hele buna birde Commodore firması battığı zaman ortadan kaybolan Amiganın özel çiplerine ait dökümanların yokluğuda eklenince çok uzun bir süre Amiga özel çipleri üretmek tatlı bir hayal olarak kaldı. Son birkaç yıl içerisinde ise, yeni bir işlemci yapısıyla karşılaştık. FPGA (Field-programmable gate array) yani basitçe programlanabilir çip olarak adlandırabileceğimiz işlemciler sayesinde, istediğiniz çipi evinizde üretmeniz bile mümkün. Eski birçok oyun konsolunun ya da bilgisayarın, günümüzde yeniden üretilmesi, joysticklere sığan küçük yapıları hep FPGA'ların sayesinde oldu. Bu gelişme Amiga içinde bir ümit oldu. Özellikle elimizdeki bilgisayarların arızalanması ve giderek işlemcilerin yok olduğu bir ortamda, yeniden Amiga üretilmesi, hatta günümüz bilgisayar dünyasının kullandığı giriş/çıkış birimlerini entegre ederek, günümüz ihtiyaçlarına daha uyan, daha kolay kullanılan bir Amiga üretmek artık mümkün. 2007 yılında duyurulup, 2008 yılbaşında satışa sunulan Minimig (MiniAmiga) bunun en güzel örneği. Amiga500 uyumlu minimig MMC karttan okuduğu .ADF dosyaları çalıştırıyor, PS/2 konnektörüyle klavye ve mouse bağlamanızı sağlarken, titreşimsiz görüntüyü (flickerfixer) destekliyor. Eksiği yok mu ? Muhakkak eksikleri, geliştirilmesi gereken yönleri var. Fakat ürünü satışa sunan Acube firmasının stoklarının kısa sürede tükenmesi, bu tür projelere Amiga severlerin desteği ve ilgisini gösterir nitelikte. Çalışmaları süren bir diğer proje olan CloneA yine minimig gibi Amiga500 uyumlu bir bilgisayar olacak. Peki daha fazlası? AGA işlemci setine sahip bir Amigayı aynı şekilde üretmek mümkün değil mi?

## Natami (Native Amiga)



**Şekil 1.**

Minimig ve Clone-A projelerinin ortaya çıkmasının ardından, FPGA kullanarak Amiga'nın en gelişmiş işlemci seti olan AGA'yı üretmek mümkün olabilirmi düşüncesi tüm Amiga severlerin kafasını kurcalamaya başladı. 2008 yılı başında Almanyanın Karlsruhe şehrinde düzenlenen MEKA Amiga toplantısında bir başka Amiga projesi katılımcıları ve ileri sürülen özellikleriyle tüm Amiga dünyasını heycanlandırdı.

Natami yani "Nat"ive "Ami"ga, bildiğimiz en gelişmiş Amiga modelinden daha fazla özellik ve güç sunmayı vaad ediyor. Genel olarak özetlemek gerekirse, Natami projesi hayata geçtiğinde modern bilgisayarlar gibi kullanabileceğiniz, kolayca terfi (upgrade) edebileceğiniz en güçlü Amiga modeline ulaşmış olacağız.

## Natami ne içeriyor?

Amiga serisinin en sağlam ve güçlü işlemci ailesi olan AGA 1992 yılında çıktığı zaman aslında çok daha gelişmiş bir işlemci serisi olan AAA işlemcisinin sahip olacağı özelliklerin bir kısmını içeriyordu. Belkide Amiga'nın gözden düşmesine sebep olacak en önemli sebeplerden biri olan 3D oyunların PC dünyasını sarsmasıyla, AGA eskisi kadar cazip görülmemeye başladı. Grafik altyapısı bitplane'lere dayanan Amiga, hızlı olarak realtime 3D görüntü elde etmenin oldukça zor olduğu bir platformdu. Daha sonraları Amiga CD32 için çıkarılan Akiko çipi, 3D oyunlar için çözüm sunmasına rağmen, 1994 yılında commodore'un batmasıyla, Amiga'nın özel işlemcileri tarihteki yerlerini aldılar.

Tüm bu olumsuzluklara ve aradan geçen yıllara rağmen, Natami'yi geliştiren ekip, Amiga'nın donanım tasarımının ve yapısının halen güçlü ve başarılı olabileceği düşüncesinden hareketle, mevcut Amiga işlemcilerini FPGA'ları kullanarak geliştirmeyi planladılar. Böylece hem klasik Amigalarla %100 uyumluluk sağlanırken, günümüz ihtiyaçlarına cevap verecek özelliklerde ekleniş olacak.

Natami projesinin en can alıcı yönlerinden bir tanesi AGA işlemci setinin oldukça gelişmiş sürümü olan "SuperAGA". Bu Commodore Amiga'nın AGA işlemci setiyle uyumlu ve 100 kat daha hızlı. AGA ile karşılaştırıldığında sadece hız olarak değil daha yüksek çözünürlük, daha yüksek renk derinliği ve en önemlisi 3D hızlandırıcı özellikler içeriyor.

SuperAGA

1. Tüm modlar flickerfixed (titreşimsiz)
2. Video giriş / Ses girişi
3. 1280x1024, 32bit çözünürlük
4. Çok daha hızlı Blitter (100 kat daha hızlı)
5. PLANAR modlar ve yeni renk modları 8Bit Chunky, 16Bit Hicolor, 32 Bit Truecolor
6. Antialiasing özelliğine sahip 3D hızlandırıcı

Şu anda prototip aşamasındaki Natami 68060 işlemcisine sahip bir CPU kartı kullanıyor. Geliştiriciler CPU'nun bu CPU kartı sayesinde değiştirilebileceğini ve PowerPC hatta daha hızlı işlemcilerin ileride eklenebileceğini düşünüyorlar. Sistemin diğer kısımlarıda oldukça heycan verici. Amiga'nın ses işlemcisi Paula geliştirilerek 24 Bit sese ve yeni özelliklere kavuşurken, orjinal Amiga tasarımıyla uyumlu olacak. PCI veriyolu, PS2 ve USB ve aynı zamanda Gigabit Network desteğiyle Natami Amiga severlerin günlük işlerinde kullanabileceği bir yapıyı vaad ediyor. Kısacası günümüz en hızlı bilgisayarı ve teknolojisiyle yarışacak olmasada, Amiga severleri fazlasıyla tatmin edecek, kullanılabilir bir Amiga.

Projeyi geliştiren ekip, ilk Natami boardlarının 2008 yazına yetiştirilmesini ve yazılım geliştiren ve Amiga yazılımlarını bu sisteme uyarlamak isteyecek üreticilere dağıtılmasını planlıyorlar. Böylece, Natami hayata geçtiğinde, günlük işlerde kullanılabilecek yazılımlar hazır olacak. Donanımın üretilmesi ise bir başka sorun. Herhalde en büyük sorunlardan biride fiyat olarak Amiga kullanıcılarının karşılayabileceği bir fiyat. Bu konuda birşey söylemek için oldukça erken. Fakat takip edebildiğimiz kadarıyla ilk destek Efika ve PegasosPC üreticisi Genesi'den geldi. Başlangıç olarak zamanında satın aldıkları yüzlerce Amiga OS lisansını projeyi desteklemek amacıyla verebileceklerini açıkladılar. Eğer projeyi geliştiren ekiple anlaşabilirlerse, Genesi yetkilileri, boardu üretmeyi ve pazarlama konusunda da destek verebileceğini, hatta Freescale Technology Forum 2008 [http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=052539903635483A95]'de natami'nin tanıtımını yapıp daha çok destek elde edebileceğini söylüyorlar.

Kafalarda birçok soru var? Gerçekten başarılı olacak mı? Hızlı ve özellikleri gelişmiş bir Amiga mümkün mü? Bunların hepsinin cevabını sanırım 2008 yazında alacağız. Ama şurası kesinki, vaad edilenler Amiga severlerin yıllardır hayallerini kurduğu Amiga tasarımına uyuyor. Amiga için çok hayaller kurduk, çok ümitlendiğimiz günler oldu, bir çoğu hayal olarak kaldı. Umarız bu sefer, bu hayal gerçek olur.

http://www.natami.net/

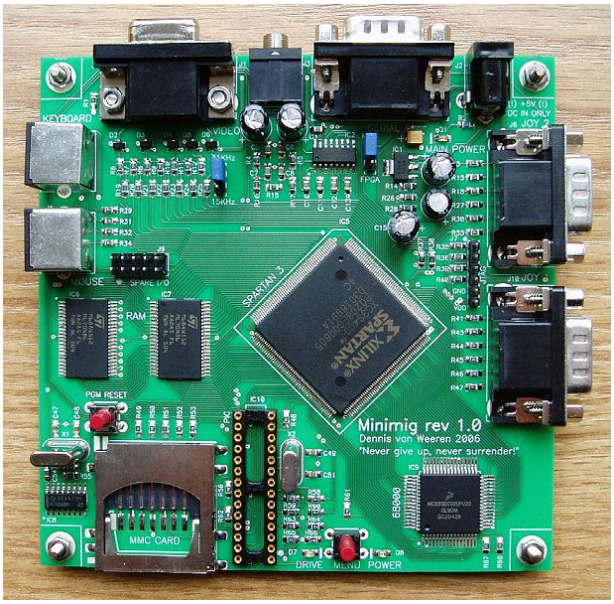
FPGA, Minimig ve CloneA projeleriyle ilgili olarak Arda karaduman'ın yazısına göz atmanızı öneririm.

# Lorraine'den MiniMig'e Amiga

Arda (CoZe) Karaduman

Merhaba arkadaşlar, bu yazıda size son günlerde adını sıkça duyduğumuz Minimig projesinden ve kısaca FPGA teknolojisinin günümüz retro sistemlerine getirdiği yeni olanaklardan bahsedeceğim.

Öncelikle Minimig projesini tanıyalım. Minimig projesi, ilk Amiga çipseti olan OCS çipsetinin bir FPGA üzerinde gerçekleştirilmesi esasına dayanıyor. Bu sistemde, disk işlemleri ve SD kart sürme işlemini PIC cinsi başka bir çip üstleniyor. Toplam 2 megabayt hafıza, 512K ROM, 512K Fast RAM ve 1mb Chip RAM olarak ayarlanmış. Boot esnasında PIC, SD Kart üzerindeki FPGA konfigürasyon dosyasını FPGA'e, ROM dosyasında kendine ayrılan 512k hafızaya yüklüyor. 68000 işlemcinin yerini ise 3,3v uyumlu 68SEC000 almış. Bu işlemci 68000 ile yüzde yüz uyumlu olmakla birlikte bir komut (move sr,EA) sadece privileged modda çalışıyor. Stack register'ı okuyan bu komutun Minimig uyumluluğu üzerindeki etkisi şu anda tartışılan bir konu olduğu için fazla bahsetmeyeceğim, ama şu an için 68020 ve üzeri işlemcileri tanımak için bu komutu kullanan programlarda sorun yarattığı düşünülüyor.




Minimig Revizyon 1

**Şekil 1.**

Minimig'i son kullanıcılar için önemli kılan en büyük özelliği günümüz donanımlarıyla uyumlu bir şekilde çalışabilmesi. PS2 mouse ve klavyenizi herhangi bir çeviriciye ihtiyaç duymadan Minimig'inizde kullanabilirsiniz. Ayrıca 15 KHz ve 31 KHz görüntü modları sayesinde klasik Amiga monitörleri dışında normal bir VGA monitör veya LCD ile de kullanabilirsiniz. SD kart yuvası ve adf emulasyonu artık tarihin tozlu rafları arasında yerini alan floppy teknolojisinden bizi kurtarıyor. Yanlızca tek bir birimde Amiga standartlarına sadık kalınmış, Joystick. Emektar Quickshot'ınızı Minimig ile kullanabilirsiniz. Ayrıca port 1 için mouse ile cebelleşmenize gerek yok, mouse portu ayrı bir ps2 port olduğundan iki joystick ve mouse u aynı anda bağlayabilirsiniz.

Peki Minimig'i geliştiriciler açısından önemli kılan ne? Öncelikle şimdiye kadar yapılamaz denilen Amiga custom çipset emulasyonunun yapılabilir bir proje olduğunu ispatlaması. Bu açıdan birçok projeye yol gösterebilecek bir gelişme. Şu an geliştirilmekte olan iki ayrı Amiga FPGA projesine umutla bakmamızı sağlıyor. Bunlardan ilki, Individual Computers'ın Clone A projesi. Bu projede Minimig gibi A500 özelliklerinde bir ECS çipset emule etme esasına dayanıyor. Ama Minimig'den farklı olarak Clone A, custom çiplerin birebir replikasını üretip oradan tüm Amiga emulasyonunu gerçekleştirmeyi hedefliyor. Geliştirme gerçek Amiga 500 üzerine yerleştirilen FPGA çipleri ile yapılıyor. Bu açıdan custom çip emulasyonunun yüzde yüz uyumlu olması planlanıyor. Diğer proje ise son zamanlarda adından söz ettiren ve oldukça spekülasyon yaratan NatAmi projesi. Bu proje AGA çipsetli bir Amiga'yı emule etme hedefini taşıyor ama aynı zamanda SuperAGA adında gelişmiş bir mod barındıracağı söyleniyor. Bu proje şu aşamada biraz belirsiz bir durumda olduğundan fazla bahsetmeyeceğim.


Clone-A Prototipi

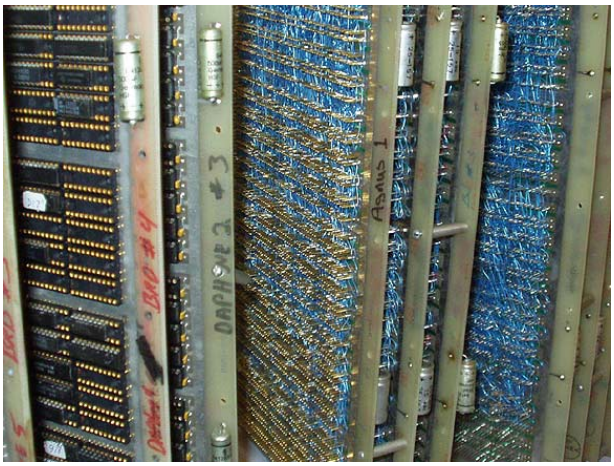
**Şekil 2.**

Minimig sadece Amiga çipseti veya başka bir retro platform emulasyonu değil, bu platformlarla kullanabileceğimiz çeşitli ara-

birimler içinde ilham kaynağı oluyor. Buna örnek olarak Amiga 500 de kullanılan IDE harddisk projesini ve Amiga 600 için geliştirilen RAM projesini verebiliriz. Gerçi bu projeler FPGA değil CPLD tipi başka bir tür programlanabilir çip kullanıyor ama kullanılan mantık aynı. Her iki projede Verilog ve VHDL gibi donanım tasarım dilleri vasıtasıyla Amiga kullanıcıları tarafından gerçekleştirilen projeler.

FPGA ve CPLD'lerden söz açılmışken biraz bu çipler hakkında bilgi vermek istiyorum. FPGA ne demek ? İngilizce açılımı Field-Programmable Gate Array olan bu çiplere Türkçe anlaşılabilir bir isim vermek biraz zor, ama programlanabilir çip diyebiliriz. Tabi böyle diyince yine kafada soru işaretleri oluşabilir, programlanabilir çip ne demek ? normal cpu'larda programlanmıyor mu? diyebilirsiniz. Ama burada bahsedilen programlama biraz farklı. Bunun farkını kavrayabilmek için biraz alt seviyeye inip ASIC olarak ifade edilen (belli uygulamalara özgü) çiplerin yapısını anlatmamız gerekiyor. Bu çipler belirli sayıda mantık kapılarının (and, or, not ve bunların kombinasyonları) dizimiyle oluşturuluyor. Ve üretildikten sonra bu yapı sabit kalıyor. Bu ne demek, yani bir 68000 işlemci üretildikten sonra ancak bir 68000 işlemci işlevini görüyor veya bir mpg decoder çipi sadece mpg çözüm işleminde kullanılabiliyor. Yeniden programlanabilir çipler ise kullanıcı tarafından konfigürasyonu değiştirilebilen mantık kapılarından oluşan bir yapıya sahip. Yani and/or/not kapıları ve bunların kombinasyonları yeniden programlanabilir çiplerde yeniden ayarlanabiliyor ve çip tamamen farklı bir işlev görebiliyor.

Peki bu FPGA'ler ne için kullanılıyor? Ne işe yarıyorlar? Öncelikle hardware tasarımı konusunda bir devrim anlamına geliyor, çünkü eskiden çok büyük ve hantal transistor plakalarıyla yapılan çip geliştirme işlemini, bir kibrit kutusu büyüklüğünde bir çip ile evde masa üstünde yapabiliyorsunuz. Mesela Amiga çipsetinin ilk tasarım aşamasında kullanılan sisteme bir bakalım :



Daphne ve Agnus Plakaları

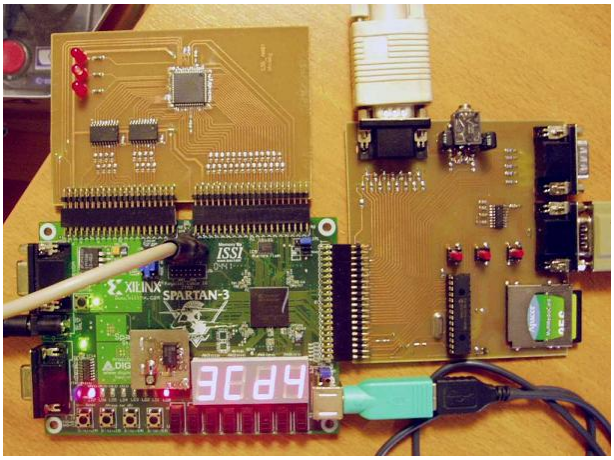
**Şekil 3.**

Bu fotoğrafta gördüğünüz sağdaki üç plaka agnus çipini oluşturuyor. Soldaki plakalar ise ilk Denise prototipi olan Daphne'ye ait. Dizaynda bir değişiklik yapmak gerektiğinde, bu plakalar üzerinde el ile gerekli değişiklik yapılıyor (atıyorum bir and or ile değişiyor, veya 85'inci and kapısının çıkışı 37'inci xor'un girişine bağlanıyor, gibi). FPGA'lerde ise bu tip işlemler tamamen software ile birkaç tuş darbesiyle yapılan bir değişikliğe dönüşüyor. HDL (hardware design language) denilen donanım tasarım dilleri ile yapmak istediğiniz değişikliği yapıp çipinize yüklüyorsunuz.

İlk Amiga Prototipi Lorraine ve 'custom chipset'

**Şekil 4.**

Burada ufak bir parantez açarak sizi 'Donanım tasarım dilleri' konusunda kısaca bilgilendirmek istiyorum. İlk donanım tasarım dili olan Verilog aslen 1985 yılında Gateway tarafından geliştirilmiş olan bir dil. Verilog'un en büyük alternatifi olan VHDL ise 1987 yılında ABD savunma bakanlığı tarafından geliştiriliyor. O dönemlerde programlanabilir çip teknolojisi fazla gelişmiş olmadığından bu diller daha çok simulasyon ortamında kullanılıyordu. İlk FPGA lerin tasarımı Xilinx tarafından 1984 yılına dayanmasına rağmen ev kullanıcılarına hitap eden Spartan serisi FPGA'lerin geliştirilmesi 1998'i bulur. 2005'e gelindiğinde ise Spartan serisi 128 milyon satan, Xilinx'e yılda 1.2 milyon gelir getiren dev bir sektör haline gelmişti. 2005 mart ayında Xilinx Spartan3E serisini piyasaya sürdü. Tam bu sıralarda ise Hollandalı Dennis Van Weeren, biraz da Commodore One projesinin etkisiyle Amiga çipsetinin FPGA üzerinde emule edilip edilemeyeceğini düşünüyordu. Aynı sene içinde kendisine bir Spartan 3 başlangıç kiti alarak Verilog ile ilk kodlarını yazmaya başladı.




Minimig Prototipi

**Şekil 5.**

Minimig prototipi ilk defa 2005 kasım ayında boot etti. Resimde görünen prototipte 68SEC000 çip ve 2mb SRAM ayrı bir PCB'de, SD kart, mouse, joystick ve video çıkışları ayrı bir PCB'de görülüyor. Bu aşamada çipset emulasyonunda birçok bug bulunuyordu. Bunların ayıklanması ve ilk versiyonun GPL lisansı altında yayınlanması 2007'yi buldu. Her ne kadar en son sürümünde bir takım buglar bulundursa bile bu projenin bu aşamaya kadar gelmesi Amiga çipsetini yeniden tasarlama amacı taşıyan projeler için çok umut verici. Amiga çipseti zamanında Jay Miner'ın önderliğinde her biri kendi alanında uzman 30 profesyonel tasarımcının çabalarıyla gerçekleşmişti. Aynı işlemi (gerçi biraz kopya çekerekte olsa) tek başına bir hobi tasarımcının gerçekleştirmesini Xilinx'in yarattığı bu FPGA devrimine borçlu olduğumuzu söyleyebiliriz.

Minimig projesinin ilk yan ürünü Tobias Gubener [http://www.opencores.org/people.cgi/info/tobiflex] tarafından geliştirilen tg68k projesi oldu. Bu proje, Amiga çipseti olarak tamamen Minimig kodu üzerine dayanmasına rağmen, 68000 işlemci yerine softcore (yani FPGA tarafından emule edilen) bir 68000 işlemci kullanıyor. Bu yaklaşım, 68000 emulasyonundan doğan uyumluluk sorunları yaratma potansiyeli olsa bile, beraberinde getirdiği yeni olanaklar sayesinde TG68'in çok ilginç bir proje olmasını sağlıyor. Bu proje ile 68000'ı daha hızlı emule ederek hızlandırılmış bir Minimig elde etmek mümkün. Aynı zamanda 68030-68060 gibi artık üretilmeyen işlemcileri FPGA ortamında gerçekleştirmeninde yollarını açıyor. Belki ileride çeşitli Amiga modelleri için FPGA bazlı turbo kartlarda görebiliriz, kimbilir ?


Altera DE2 Softcore Minimig

**Şekil 6.**

Sonuç olarak FPGA teknolojisinin gelişmesinin bizlere yeni donanım olanakları olarak geri döneceğini söyleyebiliriz. Burada bir yazının daha sonuna gelmiş bulunduk, umarım bir sonraki Plazma yazısında gelecek vaat eden bir diğer FPGA implementasyon projesi olan NatAmi hakkında daha ayrıntılı bilgi verebiliyor olacağız. Bu yazıda bahsi geçen projeler hakkında daha detaylı bilgi sahibi olmak isterseniz e-mail adresimden bana ulaşabilirsiniz.

coze.00@gmail.com

Not : Bu yazıda kullanılan fotoğraflar, Dennis Van Weeren'in Minimig sayfası, Secret Weapons of Commodore, Total Amiga Jens Schönfield röportajı ve TG68 proje grubu'ndan alıntıdır.

Linkler:

Minimig projesi resmi sayfası:

http://home.hetnet.nl/~weeren001/

TG68 yahoo grubu:

http://www.opencores.org/projects.cgi/web/tg68/overview

Minimig amiga.org topiği:

http://www.amiga.org/modules/newbb/viewtopic.php?forum=8&topic_id=27988

NatAmi resmi sayfası:

http://www.natami.net/

Clone-A röportajı:

http://www.totalamiga.org/files/TA25_JensIviewExtract.pdf

Xilinx Spartan 3 Geliştirme Kiti:

http://www.xilinx.com/products/devkits/HW-SPAR3-SK-UNI-G.htm

Altera DE2 Geliştirme Kiti:

http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=30

# MossyCon Commodore Buluşması

Bilgem (Nightlord) Çakır

ACUG (Anything Commodore User Group) tarafından düzenlenen MossyCon Commodore buluşması, Oregon eyaletinin küçük ve güzel bir kasabası olan Astoria'da, 16 Mart 2008 tarihinde gerçekleşti. Pazar sabahı erken saatlerde, benim de üyesi olduğum ve yeni dirilen UPCHUG (University Place Commodore Home Users Group - Seattle çevresindeki Commodore kullanıcalarının oluşturduğu grup) üyelerinden biri olan yeni arkadaşım Gene Woods'u da alarak Astoria'ya doğru yola koyuldum. Solda Güneş yükseliyordu... Güney'e giderken... :D

## İlk Saatler

Yaptığımız yaklaşık 3 saat süren araba yolculuğu, hem tatlı Commodore sohbetinden, hem de etraftaki doğanın harika olmasından ötürü çok güzel geçti. Hele eyaletler arası otobandan ayrılıp, Astoria'ya doğru ilerledikçe doğa daha da güzelleşti. Sonunda 11.30 civarı, buluşmanın gerçekleşeceği pizzacıya vardık.

Mekana ilk geldiğimizde, Kuzey Amerika'daki hemen her Commodore'cu için tanıdık bir yüz olan Robert Bernardo bizi karşıladı. O sırada Robert, bir dolu ilginç donanımı, sergilemek amacıyla masalara dizmişti bile. Bu ilginç müzelik objeler arasında, Jack Tremiel imzalı bir C-64, Bill Herd imzalı bir C-128, Dave Haynie imzalı bir Amiga 4000 ve ilginç olmak için kimse tarafından imzalanması gerekmeyecek kadar ender rastlanan bir cihaz olan bir Commodore PET vardı.

Buluşmaya gelen herkesin mekanda toplanması uzunca bir zaman aldı, ama sonunda birkaç saat içinde herkes geldi ve pizzalarımızı söyleyip muhabbet etmeye başladık. Bu arada ben bir yandan yapacağım sunum için hazırlıklarımı tamamlamaya çalışıyordum. Son iki yılda geliştirdiğim, Cepp isimli “6502 işlemcisi için C++ derleyicisi” hakkında bir sunum yapacaktım. Ne yazık ki, kodun son versiyonunu evdeki Linux makinemden, Windows laptop'ıma son anda aktarabilmiştim (Kod evde bir Linux build makinesinde, gecelik otamatik test ve build alt yapısı üzerinde geliştiriliyor. Bu aslında önemli bir teknik konu. Plazma'nın 3. sayısında “Yazılım Geliştirmede Faydalı Uygulamalar” adlı yazımda bu konulara değinmiştim). Neyse ki, yaklaşık yarım saat içinde kodu Windows üzerinde de derlemeyi başarabildim. Ayrıca oluşan windows cepp.exe dosyası ile, C++ kodu derlemeyi de hemen bir denedim. Bunların çalıştığını gördükten sonra muhabbete katıldım. İlerleyen saatlerde sunumumu yaptım. Kısaca Cepp derleyicisini tanıtıp, bu derleyici kullanarak yapılan Cepptris adlı basit oyunumu gösterdim. Cepptris tahmin edebileceğiniz üzere bir Tetris klonu. Bu sunumla beraber bütün sunumlar, Robert Bernardo tarafından kameraya alındı. Ardından bu video ACUG üyelerinden Andrew Wiskow tarafından internete kondu (http://cottonwood.servebbs.com/acug0447/mc4)

Katılımcılar ilk saatlerdeki tanışma ve muhabbet ile meşgul

**Şekil 1.**

## SID'in marifetleri

MossyCon'daki en ilginç katılımcılardan biri Seattle'dan gelen Steve Jones idi. Steve, mekana modifiye edilmiş bir SX-64 (Osiloskop görünümlü taşınabilir C-64) ve bazı ses ekipmanları ile geldi. Bir süre sonra anladık ki, Steve meğer profesyönel bir müzisyenmiş ve bu ekipmanı ile ürettiği otantik SID seslerini yaptığı müzikte kullanıyormuş. Sunumuna önce SX-64'üne yaptığı donanım modifikasyonlarını tanıtarak başlayan Steve, ardından kullandığı Prophet 64 adlı müzik programının çeşitli bölümlerini teker teker anlattı. Yaptığı modifikasyonlar arasında, reset devresi (ki bunu sahnede canlı performanslarda kullanıyor anladığım kadarı ile), stereo SID (yani ilave ses çipi) ve ses çıkış katındaki analog amfi devrelerinin kondansatörlerine yaptığı değişiklikler vardı (bu değişiklikler ses çıkış anfilerinin filtre karakteristiklerini daha dengeli hale getirmeye yarıyor). Bu sistemin çıkışını da bir mikser ve iki adet efekt pedalına bağlayan Steve'in elde ettiği son ses kalitesi karşısında dibim düştü. Resmen inanamadım.

Ayrıca Steve'den öğrendiğim kadarı ile, Prophet 64 yazılımı aslında kullanıcı arabirimi olarak pekçok fikri, 80'lerde çok populer olup sonra 90'larda bulunmaz hint kumaşına dönen ve fiyatı çok artan bazı ticari synthesizer'lardan alıyormuş. Ses olarak da bunlara yakın performans göstermesi yüzünden, modifiyeli bir SX64 ve Prophet 64 yazılımı, sonuçta ekonomik olarak çok daha hesaplı ve makul bir çözüm alternatifi olarak ortaya çıkmış. Elektronik müzikle uğraşanların dikkatine :) Umarım birgün Steve'i sahnede canlı izleme şansını bulurum. Bilgili ve yaratıcı bir arkadaş. Onu aylık UPCHUG toplantılarına getirmeye çalışa-

cağım.

# Dungeon Master

Diğer bir ilginç ziyaretçi de (aslında buluşmanın organiztörlerinden olması bakımından ev sahibi de diyebiliriz) Lord Ronin idi. Tam da hayatta bol bol beraber takılıp çok eğlenilecek, ve çok şey öğrenilecek türde bir adam. Yerleşik bir alaycılık (rahatsız edici olmayan türden) ve zeka izlerini hemen görmek mümkün. Benim gibi farklı bir kültürden gelen birisi için, takip etmesi zor olan pekçok TV ve kitap referansını konuşmasının arasına cömertçe serpiştiriyor olması, beni sürekli olarak sonra gidip araştıracağım çeşitli yazarlar ve yapımlar hakkında notlar almaya itti. Sonuçta benim için biraz zor ama fikir tazeleyici bir deneyim oldu Lord Ronin ile sohbetlerimiz.

Robert Bernardo'nun sergilediği envai çeşit Commodore cihaz

**Şekil 2.**

Tam bir FRP hastası olan Lord Ronin (sanırım 1976dan beri oynuyormuş, bu arada meraklıları bilir, Lord Ronin AD&D versiyon 1'cilerdenmiş) bu etkinlik için bir diğer ACUG üyesi olan David ile beraber bir CD hazırlamış. CD içinde çeşitli FRP senaryoları, bilim kurgu hikayeleri ve eski bir text adventure yapım kiti ile hazırlanmış, senaryosu yine kendisine ait olan birkaç tane adventure oyunu var. Gelenlere bedava dağıttıkları bu CD'yi henüz inceleme şansım olmadı, ama umarım Plazma'nin bu sayısının çıkmasından sonra oturup keyifle o hikayeleri okuyup oyunları oynayabileceğim. Belki gelecekte Lord Ronin'in bazı senaryolarını adventure oyunları olarak (yani text değil tamamlanmış grafik ikon adventure oyunları olarak) yayınlama projelerine girişebiliriz.

# Diğer Muhabbetler

Ayrıca David ve Rick (Wildstar) isimli iki başka ACUG üyesi ile daha tanıştım. Kendisi bir grafiker olan Wildstar ile demoscene üzerine bol bol sohbet ettik. Amerika'da demoscene hakkında yüksek bir farkındalığa sahip birilerine rastlamak ender bir durum. Çünkü buradaki Commodore'cu arkadaşlar genelde ağırlıklı olarak PAL uyumlu olan demoları, NTSC makinelerinde izleyemedikleri için bir kopukluk oluşmuş.

Daha sonra Robert bir grup değişik yazılım ve donanımı gösterdi. Bunlar arasında Super CPU destekli Shoot'em Right oyunu Metal Dust, bazı VIC-20 kartuşları ve kendi ilginç analog joystick'i ile gelen bir model uçak simulatörü vardı. Bazen burada öyle niche Commodore alakalı ürünlere rastlıyorum ki hayretler içinde kalıyorum. Yani birileri kalkmış “Abi süper bir fikrim var. Şimdi bir oyun yapalım standart dışı bir joystick gerektirsin, hem joystick hem oyun satarız. Süper” falan demiş. Yani öyle çok aksiyonlu, atraksiyonlu bir oyun da değil. Ama Commodore, o dönemde o kadar yayılmış ve pazara hakim olmuş ki, insanlar zaman zaman böyle uç uygulamalara bile girmekten kaçınmamış. Biz Türkiye'de disket sürücüye bile niche bir cihaz gözüyle bakıyorduk o yıllarda.

Sonunda akşam 7.00 civarı bizim için ayrılma vakti geldi. Yine üç saatlik bir dönüş yolculuğu bizi bekliyordu.

MossyCon benim için güzel bir deneyim oldu. Cografik olarak yakın bölgelerde oturan bu kadar Commodore'cu ile hepimize faydalı olacak yeni projelerin gelecekte olabileceğini umuyor ve görüyorum. Özellikle Seattle çevresinde UPCHUG'un da yeniden aktive olması ile bu bölgedeki kullanıcılar arasında ne tip bir üretkenlik ve dayanışma yaratabileceğimizi merakla izliyor olacağım.

# Forever 9 Parti Raporu

## Arnold (Jailbird) Cistai

Bu senenin başlarında, 2008'in ilk yarısında ziyaret etme şansım olan iki partiyi merak etmeye başlamıştım. Bunlardan ilki benim ana hedefim olan Breakpoint idi (birçok bilindik nedenden ötürü) ve diğeri ise benim açımdan ucuza gelecek ve ulaşılması çok kolay olan Forever 9 partisiydi. Her iki aktivitenin arasında yalnızca bir haftalık bir zaman dilimi vardı. Bir sure boyunca Slovakya yolculuğunu yapıp yapmama konusunda teredütte kaldım. Ama ikinci kez düşündüğümde bu çılgın Macar partisinin kaçırılmaması ve kesinlikle orda olmam gerektiğine kanaat getirdim.

Benim kayınpeder uzun bir aktörlük kariyerinin ardından bir ulusal Macar ödülü aldı ve Cuma öğleden sonra yapılacak törene davet edildi. Ben de Poison/Singular ile buluşacağım yere, Budapeşte'ye beni yanında götürmesini rica ettim. Sonradan bunun büyük bir hata olduğu ortaya çıktı. :)

Öncelikle yolculuğumuza oldukça geç başladık. Kayınpederim biraz hızlanarak arayı kapatmaya karar verdi ancak Murphy'nin kanunları bizi vurdu: polis tarafından durdurulduk. Polise kibar davranıp bir an önce siktir olup gitmek yerine polisle tartışıp bağırmaya başlayınca 20 dakika kadar orada sıkışıp kaldık. Zaten çok gecikmiştik ki Murphy'nin kanunları ikinci kez uygulandı: sınırda oldukça uzun bir araba kuyruğu oluşmuştu. Her zaman zeki takılan sevgilimin babası, trafikten yırtmak için emniyet şeridini kullanmaya karar Verdi. Macar başbakanının imzaladığı bir davetiyemiz vardı ve bir şekilde bu davetiyi sınır devriyelerinin burnuna sokmak işe yaradı, sınırı geçmeyi başardık, hem de oldukça hızlı bir biçimde.

Budapeşte'ye kadar oldukça hoş bir yolculuk yaşadık, ama şehre geldiğimiz gibi yoğun bir trafikle karşılaştık. Burnumuza garip bir koku geliyordu. İlk başta dışardan geldiğimiz bu kokunun çevredeki insanların bağırarak bizi uyarması üzerine arabamızdan dökülen bir sıvıdan geldiğini anladık. Murphy'nin kanunları 3. kez işliyordu. Zaten çok gecikmiştim. Bu noktada kısa bir vedalaşmanın ardından kendimi arabadan dışarı atıp, herhangi bir toplu taşıma bulunan en yakın istasyona doğru koşmaya başladım. Trenimin kalkmasına 1 saat kadar bir sure kalmıştı ve ben hala şehrin bir ucundaydım. Böylece elimde bagajlarla hiçbir araca çarpmamaya çalışarak yoğun öğle trafiğinin içinde, araçların arasında koşuşturmaya başladım. Ama sonuç olarak trenimin kalkmasından once istasyona yetiştiğim, hatta hatta Çin restoranında sersem bir garsondan bir bira içecek vakit bulduğum için kendime gurur duyuyorum.

Poison ve ben, Macaristan-Slovakya sınırına kadar küçük bir yolculuk yaptık. Burada Singular grubundan şehrin batı bölümünden arabayla gelecek olan Cargo, Soci ve Leon'la buluştuk. Benzin istasyonundan daha da bira aldıktan ve depoyu doldurduktan sonra beşimiz Trencin yolunu tuttuk.

Slovakya sınırına yakın bir bölgede Poison ve Leon az once aldıkları güzel biralarını tadıp, kameraya poz verirken.

**Şekil 1.**

Aceleci şöförümüz Cargo'yu biraz sinirlendiren sayısız işeme molasından sonra, akşam 6 sularında parti mekanına vardık. Trencin, Slovakya'nın kuzeyinde kalan küçük ama çok hoş bir şehir. Şehir temiz ve rahat, kızları güzel ve her ne kadar İngilizce bilen birini bulmak çok zor olsa da şehrin sakinleri hoş ve yardımsever insanlardan oluşuyor. Dolayısıyla insanlarla sınırlı bir biçimde anlaşabildik. Genellikle işaret dili ya da Slavik dillerin yakınlığından medet umarak benim Sırpça konuşma numaramla anlaşmaya çalıştık. Bulunduğumuz yer şehir merkezine yarım saat mesafede, benzerlerinden çok daha kullanışlı bir büyüklüğe sahip bir kültür kulübüydü. Herkes mekanda bulunan motifleri sevdi. Parti alanı çok genişti, güzel bir lobisi vardı ve scenerlar aynı zamanda binanın dışında da takılıyorlardı. Aynı zamanda parti mekanının yanı başında, Cuma günü death metal konseri ve Cumartesi günü de punk konseri olduğunu hatırlatmalıyım. Bu sayede aktif olarak yerel underground atraksiyonlara katılıp, farklı ritüelleri deneme şansını elde ettik. :) Tek negatif nokta mekanda duş olmamasıydı. Ben bir şekilde ıslak mendillerle idare ettim parti boyunca ancak uzun yolculuğun ardından en azından bir duş alabilmeyi çok isterdim.

Biz mekana ulaştığımızda parti zaten kalabalıklaşmıştı. Çoktan mekana gelmiş bulunan C64 çetesini selamladık. Eşyalarımızı yerleştirdikten sonra gelenekselleşmiş Cuma gecesi aktivitesi olan gece kulübüne gidip, ölmeye yakın bir duruma gelene kadar içerek 8-bit parti ruhunu hissetme olayına girdik. :) Ne yazık ki bu defa yalnızca Poison yerel tatlı ama sert içeceklerle birkaç güzel birayı karıştırmanın büyük yardımıyla bu görevi tamamlamayı başardı.

Cuma gecesi içki muhabbeti: Jailbird, Cargo, Sad ve Poison (fotoğrafı çeken Leon)

**Şekil 2.**

Gecenin ilerleyen saatlerinde Comankh/Agony Design ve kız arkadaşıyla tanıştık. Bize yiyecek alabileceğimiz mekan bulmak konusunda yardımcı olmaya çalıştılar ancak sonuç olarak kaybolduk. Bir şekilde Cargo ve ben diğerlerinden ayrıldık ve ikimiz yiyecek içecek birşeyler bulana dek şehirde boş boş dolandık durduk. Şans eseri birkaç saat içinde parti mekanına dönmeyi becerdik ve bu günü bu şekilde sona erdirmeye karar verip uyuma kararı aldık.

Cumartesi, Cargo, Leon ve ben yarışmalar başlamadan önce, sahil şeridinde bir şehir turu yapma kararı alarak, parti mekanını terk ettik ve bulunduğumuz yere yakın olan güzel Trencin Kale'sine doğru yola koyulduk. Rehberimizi beklemek istemediğimiz için kale içinde kendi kendimize boş boş dolanmaya ve kendi eğlencemizi yaratmaya başladık. Örneğin insanları elimizdeki kestanelerle vurmaya çalışmak ya da mini etekli piliçleri dikizlemek gibi. Hava süper olduğu için karşılaştığımız ilginç herşeye bakmak için uzun uzun zaman harcadık. Geri dönüşte, gündüz gözüyle görmek için şehir merkezini üzerinden geri dönmeye karar verdik ve dönüş yolunda bir alış veriş merkezine uğrayarak azalan bira stokumuzu tamamladık. Slovakya ile ilgili süper bir olay da ucuz ve güzel birası. Partiye geri döndüğümüzde insanlar yavaş yavaş yarışmalara hazırlanıyorlardı ve biz de muhabbet, yeme, içme ile zaman geçirmeyi denedik.

Sonunda yarışmalar başladı ve ben bu sene olağanüstü birşeyler görüp görmeyeceğimizi çok merak ediyordum. Ne yazık ki hiçbir platformda beklediğim üstünlükte birşey yayınlanmadı ancak yarışmalar oldukça hoştu, her kategoride temiz ürünler vardı. C64 kategorisindeki tek demo Padua'nın Shorture demosuydu ve oldukça sakin bir akışa ve güzel rutinlere sahip olan bu demo belki de tüm ürünler arasında Forever partisinde yayınlanan en iyi üründü. Ne yazık ki Speccy ve Atari demolarını çok iyi hatırlayamıyorum ama en azından iki demo ilgimi çekmeyi başarmıştı. Biri "Jozin z Bazin" isimli 70'lerde doğu ülkelerinde popüler olan komik müziklerini içeren bir youtube videosuyla ilgiliydi. Diğer demo ise birinci olan Spectrum demosuydu. Oldukça heyecan verici ve tipik Skrju sitiline sahip bir demoydu.

Trencin kalesi

**Şekil 3.**

Müzik ve grafik yarışmalarına da katılım yüksekti ve ürünler gayet iyiydi (o ana kadarki C64 ile ilişkili yarışmalarda pek birşey yayınlanmadığını düşünecek olursak grafik ve müzik yarışmalarının beni şaşırttığını itiraf etmeliyim). Saehn'ın resmi çok güzeldi ve hızlıca hazırlamış olduğum multicolor resmin ikinciliğinden oldukça tatmin oldum. Aynı zamanda belirtmek isterim ki, Agony Design grubundan eski bir C64 sceneri olan Comankh da geri dönüşünü yeni bir grafik yayınlayarak duyurdu. Umarım yakın gelecekte kendisinden daha fazla grafikler görürüz.

Müzik yarışması katılım adedi olarak grafik yarışmasına nazaran biraz daha azdı, fakat Factor6 ve A-Man'in harika müzikleri günü kurtardı, en azından benim açımdan. Ah unutmadan, partinin ana teması zombiler ve ölümden sonra yaşam olduğu için Wotnau'nun yarışmaları nekromantik (ya da bir büyücü) gibi bir kıyafetle sunması güzel bir dokundurmaydı. Gel gelelim gevelediklerinin tek bir kelimesini bile anlayamadım. :) Bir diğer ilginç nokta ise bedava Gameboy ve PS1 aksesuarları dolu büyük bir paketin bizlere sunulmasıydı. Gerçi kimse çok fazla ilgilenmedi

ama yine de katılımcılara bunları sağlayanlara teşekkürler.

Çeviren: Emir Akaydın (Skate / Glance)

Parti mekanı

**Şekil 4.**

Parti bittiği gibi biralarımız bitene kadar mekanda eğlenmeye devam ettik ve sonrasında Leon'a yakınlardaki bir bara gitmeyi teklif ettim. Hiçbir Slovak barı kahve satmadığı ve yalnızca alkollü içecekler sattığı için (ki bence çok sempatik bir şey bu), Leon'un kahve içme denemeleri sonuçsuz kaldı. Pek fazla alternatifimiz yoktu ve tek seçenek olarak daha da içki içtik. Gecenin ilk ilginç olayı bardaki erkeklerin ilgisini çekmek için birbiriyle yiyişen birkaç sarhoş kızdı. Ne yazık ki erken ayrıldılar, bar da kapanıyordu zaten ve biz daha yeni sarhoş olmaya başlamıştık. Aklımıza gelen tek mantıklı hareket eğlencemize başka bir mekan bulup orada devam etmekti. Biraz yürüyüşten sonra bir pasajın girişindeki kalabalık dikkatimizi çekti. Aradığımız yer bu olmalıydı. Birkaç biranın ardından dans pistinin yanında dururken yanımızda şüpheli birkaç adam belirdi ve sayıları gittikçe artıyordu. Ben kaşlarımı kaldırdım ama hemen arından düşündüm, dedim salla gitsin, bu yerel bir adet falan olmalı diye. Fakat iri yarı adamlar ateşli bir biçimde dans etmeye, birbirlerini okşamaya, sarılmaya, yalamaya başlayınca az daha içtiğimi çıkaracaktım. Bu, birayla yıkanmış beynimin şok geçirerek gerçekleri fark ettiği ve yanlış zamanda, yanlış yerde olduğumuzu anladığı an idi.

Parti mekanına geri döndüğümüzde partinin bitmek üzere olduğunu gördük ve diğer çocukları uyandırarak toparlanmaya başladık. Birkaç saatlik yolculuğun ardından yeniden eve dönmüştüm. Çok yorgundum ancak taze hislerle ve arkadaşlarımla geçirdiğim güzel geçirilmiş bir hafta sonunun hatıralarıyla doluydum. Forever 9 gerçekten eğlenceliydi. Yalnızca iyi organize edilmiş bir 8 bit meraklıları partisi olduğu için değil, aynı zamanda bu etkinliğe eşlik eden ilginç tecrübelerle de birlikte çok güzeldi. Seneye orada olun, kesinlikle pişman olmazsınız!

Yazan: Arnold Cistai (Jailbird / Booze Design)

# Rakı, Balık, Spectrum... ve Breakpoint 2008

Arda (Ref) Erdikmen

## Spectrum PC Sahnesinde!

Koopaville demosunun programcısı Gasman'e, "Bu yıl katılıyor musun?" diye sorduğumda, "Şu aralar speccy kodlayacak gibi değilim, ama gezmeye ve yenilerle tanışmaya geleceğim" demişti. Ama O'nu bulmak için koca salonu defalarca tavaf ederken (buluşmak için bir yer planlamayı unutmuştuk) bir sürü ünlü scener ile tanışan ben olmuştum.


*Parti sırasında hala ayık ve güleryüzlü kalabilen nadir kişilerden biri olan Gasman, beleş Scene.org şampanyasından içmek için dahi masasından kalkmadı.*

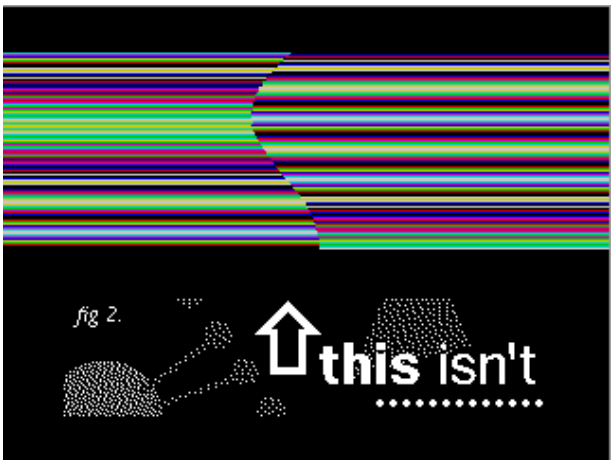
**Şekil 1.**

Evet, bu sene Breakpoint'deydim ve hiçbirşey iyi başlamadı. Vizemi partinin başlayacağı sabah alabildim ve o sırada okulda ders anlatıyordum. Akşam dersten çıkıp cuma trafiğinde uçağa son anda yetiştim ve gece yarısı Almanya'nın bomboş sokaklarında kendimi Bingen'e ulaşmaya çalışırken buldum. Son saniyede trene bindim, fakat ineceğim durakta tren kapısının bozulması üzerine inemeyerek 40 dakika sonraki durak olan Koblenz'den geri dönmek zorunda kaldım ve açılış seromonisini kaçırdım. Ardından kar yağmaya başladı ve ilk günün sonunda boğazım şişmişti. Ama bundan sonrası rüya gibiydi.

**Şekil 2.**

Breakpoint, katıksız demoscene etkinliği, bu yıl yine 1000'in üzerinde demo meraklısını bir araya topladı. Tüm anılarımı yazarsam bir kitabı doldurabilirim çünkü başından sonuna eğlence, arkadaşlık ve bira dolu 4 gün geçirdik. Ayrıca çok şey öğrendim.

"Spectrum için ise şanssız bir yıldı" demek yerine, "Bütün 8bitler için şanssız bir yıldı" diyeceğim. Aslında Zx Spectrum için bu yıl özel bir yıldı çünkü bir Zx Spectrum grubu olan "Ate Bit", PC64k kategorisinde bir Zx Spectrum demosuyla(!) birinci oldu. Hmm, bu da nesi diyebilirsiniz? Açıklayayım, Ate Bit son zamanlarda yazdıkları bir wrapper ile Zx Spectrum için çıkardıkları demoları exe'ye çeviriyorlardı. Böylece emulasyona gerek kalmadan Speccy demolarını windows pc'nizde izleyebiliyordunuz. Bu sefer Ate Bit, overclock edilmiş bir z80 işlemci kullanıyor ve tüm kodu z80 assembly ile yapıyor. Büyük olasılıkla demo tamamen zx uyumlu değil, bu sebepten bir portunu zx'de göremeyeceğiz.

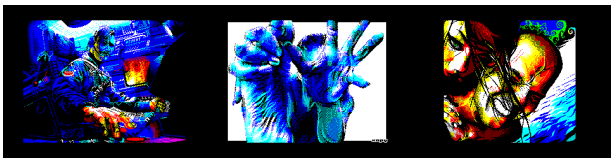



*Koopaville, ZX Spectrum'un raster yeteneğini gösteren bir demo.*

**Şekil 3.**

Yarışmalardan önce demonun yazarı EvilPaul ile iki kere ko-

nuşma fırsatı buldum. İlki parti mekanında idi. Gasman konuşurken, parmağını uzatıp “EvilPaul şurda oturuyor” diye gösterdi. Herkes kafayı çekerken o son düzeltmelerini yapıyordu. Gasman bana dönüp, “Sanırım çok heyecanlanacaksın çünkü Diver bir grafik yollayacakmış” demişti. Demo’nun PC için olacağını duyduğumda ise önce bir (tabiri caizse) yamuldum. Diver gibi bir ZX grafikeri de tamamen pc’ye mi geçmişti? Evilpaul’ün bu sene zx demosu kodlamayacağını da duymuştum, büyük hayal kırıklığıydı. (Hatırlatayım, Evilpaul tem teşekküllü bir 8bit insanıdır. Neredeyse tüm 8bit platformlarda işleri mevcuttur.) Hemen Evilpaul’ün yanına gidip elini sıktım, “Zx demosu yok mu?” dedim, “yok” dedi ve ekledi, “ama güzel olacak gibi görünüyor”. Onu fazla tutamadım çünkü başka soracak bişeyim kalmamıştı. Ertesi gün otobüs durağında tekrar sohbet ettik, bana “Meraklanma bayılacaksın” demişti, Diver’dan gelecek grafiğin de ulaştığını ve demoya yerleştirdiğini ekledi.



*Diver, özellikle “4th Dimension” grubunda yaptığı çalışmalarla tanındı.*

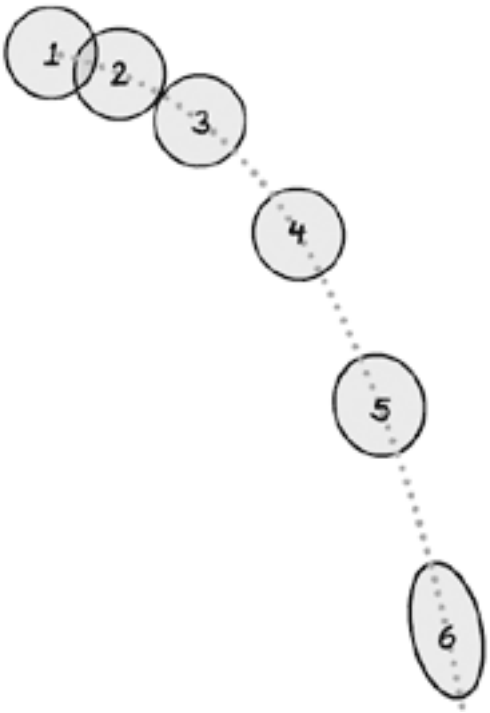
**Şekil 4.**

Ertesi gün, PC 64K demo kategorisinde parti atmosferini koklamış, izleyicilerinin nabzını tutmuş bir ZX demosu izledik. Herkesin enerjisi yerinde ve hala ayık olduğu saatlerde yapılmasından olsa gerek, tüm izleyicilerin ayakta izlediği kategori PC 64k idi. Zaten toplam 4 entry vardı ve “Pimp My Spectrum” sonunda sahnede göründü. “Hello PC scene, this is Spectrum Scene. We think we can do much better...” laflarını Sir Clive’ın konuşma balonundan okuduğumuzda dev ekranda oynayan demo ile hepimiz bir yakınlık kurmuştuk. Ustaca seçilmiş metin daha ilk baştan izleyiciyi eline almış, herkesle iletişim kurmuştu. Demo sona erdiğinde tüm parti boyunca en uzun alkışı alan ürün ünvanı “Pimp My Spectrum”a gitmişti. Bu arada eklemeden geçemeyeceğim, “Pimp My Ride” Amerika ve ingilterede popüler bir televizyon şovu. Sunucusu ise Xzibit adında bir rapper. Demonun konusu, Xzibit’in Spectrum’umuzu modifiye ediyor olmasıyla ilgili. Tv şovunun bazı klasik dialogları direkt olarak demoya geçirilmiş. Demonun sonunda Xzibit, “PC Scene, you’ve been officially pimped!” diyor. Bu cümle Ate Bit’in, PC scene’inin normal halinden, bir zx demo eklenerek daha iyiye dönüştürdüğünü anlatıyor gibi.

Zx Spectrum ile ilgili kendime başka haber çıkaramadım ama Ate Bit demosundan sonra arada sırada “Amigaaaa!” (parti anketine yapılan gönderme bir geyikti, parti boyunca sürdü) diye bağıranlara, “Spectruuum!” diye cevap geldiğini de duydum.


*Amiga Demo kategorisi başlarken, stada giren fanatik taraftarlar gibi, bir grup amigacı ellerinde bir pankart(!) ile sahneye fırladı, elbette alkışlarla birlikte tüm salonu gezerek pankartı ikinci kata astılar.*

**Şekil 5.**




*Geleneksel animasyonda motion-blur sürekli kullanılır (6 numaralı top, iyice hızlanmıştır).*

**Şekil 6.**

PC demo yarışmasını ise bu yıl Farbrausch aldı. Son yılların en etkileyici demosuydu bence. İlk gördüğümde müthiş akıcılıkla

dona kaldım. Daha ilk planda yanımda ilgiyle izleyen impetigoya dönüp "Bu yağ gibi akıyor be!" dediğimi hatırlıyorum. Gerçekten de yağ gibi akıyordu ve sebebini eve döndüğümde öğrendim. Birçok kişinin gözünden kaçmıştı ama "Masagin", animasyonu yumuşatmak için bir motionblur teknolojisi kullanıyordu! Animasyonla uğraşanlar çok iyi bilirler. Motion blur, yani devinim bulanıklığı, gerçekliği yaratmanın en önemli ögelerinden biridir. Geleneksel (kalem kağıtla yapılan) animasyonda dahi motion blur etkisini yaratacak deformasyonlar kullanılır. Böylelikle hareket tamamlanmış olur, atlamalar ya da kesilmeler azalır. Hesaplanması ise çok pahalıdır.

30 demoyu peş peşe izlediğinizde fark o kadar açıktı ki efekti farkedin ya da farketmeyin, demonun hiç teklemediğini, kare atlamadığını, gözünüzün hiç yorulmadığını görebiliyordunuz ve yarışma sonucuna da bu etki etmişti. Farbrausch demosu, "Masagin" birinci oldu. Sonuçlardan önce impetigo ile "Ne kadar cesur renk kullanımı, çok zordur bunları tasarlamak", "adamlar herşeyi okuyorlar, var olan herşeyi kullanıyorlar, kilim desenlerini falan almışlardır" diyordum ki, hem haklı hem haksız çıktım: Demodaki en önemli bölüm, Paniq'in havlusu çıktı. Ödül törenine, boynuna astığı havluyla çıktı, farkeden çok az kişi olmuştu. Ödül töreni sonrası GarbageTruck ile karşılaştım. Onun da boynunda bir havlu vardı. Beni görünce durdu; O'na, "Belki ben de senin havlundan bir demo yapmalıyım" dedim. Anlamamış gibi baktı. Tesadüf eseri Paniq biryerden belirdi ve "Bir havlu her işe yarar" dedi, Douglas Adams'a gönderme yaparak, Truck ikimize de garip garip bakmaya başlayınca, "Don't panic" diyerek ordan ayrıldım. Sonra neden hazır Paniq'i yakalamışken sıkıp suyunu çıkarmadım diye dövündüm, partinin adamı, kazanan demonun sahibini elimden kaçırmıştım. İşin içinde Neuro olduğuna göre kod python ile yazılmış olmalıydı, en azından sıkı bir tebrik etmeliydim.


*Yunan grup ASD'nin veteran müzisyeni Amusic, ödül töreni sonrası bana "En iyi demo ödülünü aldık ama ben müzik ödülünü alamadım" diye dert yandı.*

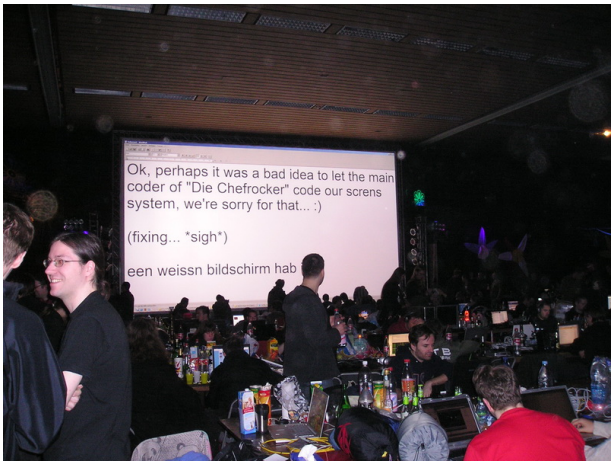
**Şekil 7.**

Bir başka önemli nokta ise Farbrausch ve ASD'nin ne kadar ileri görüşlü olduklarıydı. Bu iki grup zamanın ötesinde gibiydi gerçekten. Nedense Navis (ASD coder) ile sohbet ederken bunu ona söylemeyi unuttum ama adamlar yeni bir tür yaratıyorlardı (bunu leviathan'a söyledim, ben anlamam o işlerden ben sadece gitar çalıyorum dedi). İki grup da bu yıl iki boyutlu vektörlere yönelmişlerdi ve elbette bunun bir sebebi olmalıydı. 3D'ye bu kadar hakim olan ödüllü gruplar, neden iki boyuta dönmüşlerdi? Düşününce cevap çok basitti: HD. Yüksek çözünürlüklü (1900x1080) ekranlarda detay o kadar görünür hale geliyor ki, bunun tasarımınıza yansımaması imkansız. Mesela küçük nesneler kullanma lüksüne sahip oluyorsunuz. Üstelik bu küçük nesnelerin inanılmaz detayları olabiliyor. Kötü tarafı ise, düşük çözünürlüklü bitmap kaplamalarınızın iğrenç görünmesi. 3D objenin vektörel kısmı çok keskin ve mükemmel iken, üzerine kaplanan materyal çamur gibi birşey oluyor. Bu durumu aşmak için iki yöntem kullanabilirsiniz: yüksek çözünürlüklü kaplamalar ya da procedural kaplamalar. Birincisi çok bellek, ikincisi hem bellek hem işlemci harcar. Bu yeni yüksek çözünürlüklü ortama hükmetmenin en pratik yolu vektör kullanmak. Böylece çözünürlük sorununu otomatik olarak halledilmiş oluyor. Bu problemi aşana kadar HD demolarda kaplamalı 3d objeler genellikle kötü etki yaratacaklardır. PC 4k yarışmasını alan HD demonunda kaplamasız 3d objelerden oluştuğunuda eklemek istiyorum.

*Yunan coder Navis ile partideki tüm Türkler (impetigo, decipher, wisdom, ref=4 kişi) aynı karede.*

**Şekil 8.**

Real Wild kategorisindeki "Scheisse" (Bok) isimli kısa film ise scener'lar arasında kabul gördü. Aslında ismi gibi boktan bir filmdi, fakat kimse umursamadı, parti sonuna kadar her 10 dakikada bir, salonda "Scheisse" sesi duyuldu. Özellikle beğenilmeyen yarışma ürünlerinin izlenilmesinin hemen ardından "Scheisse" diye bağırılması eksik olmadı. Ayrıca Topy hemen t-shirtlerini basıp satmaya başladı. Laf o kadar çabuk benimsendi ki, sanırım benim kaçırdığım birşey var diye düşündüm. Büyük olasılıkla geçen yıldan kalan ("Hallo-Was?" gibi) ya da web üzerinde süregelen bir geyik olduğunu sanıyorum.




*Parti boyunca sürekli göçen sunum yazılımı screens.exe'yi yazan "Die Chefrocker" grubu üyelerine, sonunda orgalardan şaka yollu bir serzeniş, notepad aracılığı ile geldi.*

**Şekil 9.**

Hem sevindirici hem üzücü bir olay ise tanıştığım tüm scener'lar eninde sonunda Nightshift demoparty'i sordular. Partimizin dünya çapında kabul görmüş ve hatırlanan bir parti haline gelmiş olduğuna emin oldum. Birçok scener, parti olması halinde Türkiyeye gelmek istediklerini de söylediler. Malesef bu yılki Nightshift için herhangi bir duyuru yapılmadı.
emphasis>

*Fiver2, bu yıl Scene.org ödüllerinde en iyi yönetmen ödülü aldıktan sonra, Leia ile beraber kafayı çekiyor. Bu fotografı çekmek için 3 kere tekrar yapmak zorunda kaldık...*

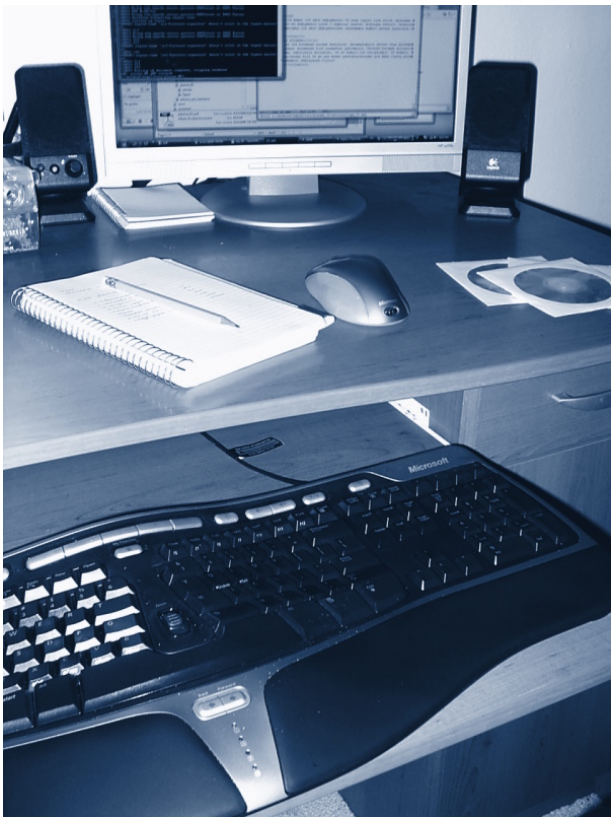
**Şekil 10.**

Beni endişelendiren bir başka şey ise scene'in özgürlükçü ve protest yapısına büyük şirketlerin el atmasıydı. Bu yıl Sony finansmanıyla ünlü Plastic grubuna yaptırılmış olan bir PS3 demosunun, yarışma dışı özel gösterimi yapıldı. Ayrıca Farbrausch'un kazanan demosu aslında bir invitro idi ve avrupalı scener'ları amerikaya davet ediyordu. Nvidia, Amerikadaki partinin sponsoru idi ve partinin adı da "Nvision" olarak belirlenmişti. Parti sonunda Intel, pc demolarını kodlayanlara özel intel lisansı hediye etti. Nvidia ise 10 gruba (ilerde geri iade edilmek üzere) Nvidia developer kit'i hediye etti. Kişisel görüşüm şöyle: bir partiye sponsor elbetteki gereklidir, ama içeriğe etki ettiğinde bu işte bir hata var demektir. Birkaç şeye sinirlendim, Sony'e demo yapmayı kabul eden plastic'e, bunu scene demosu diye yayınlayan breakpoint orgalarına, uygar dünyanın demopartisini kovboy diyarına çekmeye çalışan Nvidia ve Farbrausch'a....

Partinin başında, Gasman'in yanına oturduğumda birşeyler kodladığını gördüm, ağır ingiliz aksanının arasından zorlukla seçebildiğim kelimelerden anladığım kadarıyla sohbete dalmak yerine oturup bir fast demo yapmaya karar vermişti. Son gün bazı yarışmalardaki işlerin çok zayıf kaçtığını gördükten sonra neden ben de birşeyler yapmadım diye dizlerimi dövdüm herzamanki gibi. Ama bu parti benim için tanışma ve sohbetle dolu oldu, birdahaki sefere oraya ulaştığımda elimin boş olmamasını umuyorum. Bunu da garantilemek adına Crescent'e katıldım. Wisdom ve Impetigo'nun harika müzikleriyle çalışmak çok zevkli olacak. Umarım Türkiye'deki tüm gruplar tekrar harekete geçer (özellikle Resident ve Demodojo) ve Farbrausch'un demosundaki ülke bayrakları dizisinde Türkiye'nin de yer alması sadece lafta kalmaz.

# C++ ve Nesne Yönelimli Programlama

Bilgem (Nightlord) Çakır



**Şekil 1.**

## Giriş

Merhaba sevgili okuyucular. Bundan böyle, bu köşede dünyada en çok kullanılan programlama dillerinden biri olan C++ dilini öğrenmeye başlayacaksınız. Aynı zamanda Nesne Yönelimli Programlama'nın (Object Oriented Programming, OOP veya bazen de Object Oriented Analysis and Design yani OOAD olarak yabancı kaynaklarda karşınıza çıkabilir) ne olduğunu ve C++ dili ile nasıl Nesne Yönelimli Programlama yapacağınızı görüyor olacaksınız. Nesne yönelimli programlama yapmak için kullanabileceğiniz tek dil C++ değil. Smalltalk, Java veya C# gibi başka nesne yönelimli diller olduğu gibi, zaten son yıllarda dünyadaki bütün programlama dilleri Nesne Yönelimli yaklaşımları bünyelerine katıyor. C++ bu diller arasında ihtimalen en yüksek bellek ve CPU performanslı programları üretebilmenize imkan tanıyan çok güçlü bir dil.

Temel olarak kursumuz bu iki kulvarda paralel ilerliyor olacak. Bir kulvarda C++ hakkındaki bazı kavramları öğrenirken, diğer kulvara atlayıp Nesne Yönelimlilik ile ilgili bazı kavramlara değineceğiz. Sonra tekrar C++ kulvarına geçip öğrendiğiniz yeni bir Nesne Yönelimli yaklaşımı C++ ile nasıl kullanacağınızı göreceksiniz.

C++ dünyadaki belki en güçlü programlama dillerinden olmasına karşın malesef yazılımcılar tarafından çoğu zaman verimsiz kullanılmaktadır. C++ kullanan pekçok yazılımcının bu güçlü “dilden” defalarca “dili” yanmıştır. C++'ı tam anlayıp öğrenmeden kullanmaya kalktığınızda size de çok zor gelebilir. Ancak bu gözünüzü korkutmasın. Bu dilin bütün inceliklerini yavaş yavaş başlangıç seviyesinden alarak inceleyecek ve anlatacağız. Herhangi bir başka programlama dilini bilmiyor olduğunuzu varsayacağım. Bu yüzden de en temel kavram ve komutlardan başlıyoruz. Bildiğinizi varsayacağım iki şey var. Birincisi işletim sisteminizde dosya ve klasörleri kullanabilmek. Yeni klasör ve dosyalar yaratıp silebiliyorsaniz tamamdır. İkincisi de işletim sisteminizde konsol programını açmayı bilmeniz. Yazdığımız programları yeni yaratacağınız text tabanlı dosyalara yazıp konsoldan derleyiciyi kullanarak çalışabilir (.exe) hale getiriyor olacağız.

Ayrıca genel olarak CPU ve Bellek gibi kavramlardan da bahseden Bilgisayar Mimarisine Giriş yazı dizisini de takip etmenizi tavsiye ederim.

## Ön Hazırlıklar

Eğer Windows kullanıyorsanız ve hali hazırda yapmadıysanız yapmanız gereken bir ayar var. Hemen bir Windows Explorer penceresi açın. Üst menüden Araçlar->Klasör Seçeneklerini seçip orada “Bilinen dosya türlerinin uzantılarını gizle” seçeneğini bulun ve onu seçilmemiş hale getirin. Bu değişikliği yapınca artık Explorer'da bütün dosyaların .exe, .txt gibi uzantılarını görebiliyor hale geleceksiniz. Bu önemli değişikliği yapmazsanız sonra bazı basit hatalardan uzun vakitler kaybedebilirsiniz.

Bunun yanında kendinize bir çalışma klasörü oluşturun. Örneğin D:\kaynak veya /home/ahmet/kaynak vs gibi... Buraya kursun geri kalanında Kaynak Klasörü diyeceğim

## Derleyici Kurulumu

C++ ile program yazabilmek için bilgisayarınızda derleyici (compiler) denilen bir yazılım kurulu olması gerekiyor. Bu kursta ilerlerken hangi işletim sistemini ve hangi derleyiciyi kullandığınızın bir önemi yok.

Eğer Windows kullanıyorsanız benim tavsiye edeceğim iki ücretsiz derleyici var. Bunlar Visual Studio Express (http://www.microsoft.com/express/download/) ve Mingw (http://sf.net/project/showfiles.php?group_id=2435 buradan en tepedeki Mingw Automated Installer seçeneğini kullanın ve kurulum sırasında g++ compiler seçeneğini işaretlemeyi unutmayın)

Eğer Linux kullanıyorsanız gcc adlı derleyiciyi kullanabilirsiniz. Büyük olasılıkla zaten sisteminizde kuruludur. Yine g++ seçeneğinin de kurulu olduğundan emin olun. Bazen gcc kurulu olup g++ olmayabilir, dikkat edin.

Windows ve Linux dışındaki bütün platformlarda da ihtimalen gcc'nin bir portu vardır.

Bu bahsedilen derleyicilerden birini kurduktan sonra, bir konsol penceresi açın (Visual Studio kullanıyorsanız, Başlat->Microsoft Visual Studio->Visual Studio Tools->Visual Studio Command Prompt; gcc için ise herhangi bir konsol). Derleyici programının adını yazın (VS için “cl”, gcc için “g++”). Eğer konsol size programı bulamadığını söylerse derleyiciyi kuramadınız demektir. Bu durumda web forumlarında yardım bulup bu problemi aşmaya çalışmalısınız.

Eğer aşağıdaki hatalardan birini alırsanız herşey yolunda demektir.

```
Gcc: no input file
VS: usage: cl ......
```

Artık kursa hazırsınız.

# İlk Program

Şimdi ilk C++ programımızı yazacağız. Bunun için kaynak klasöründe yeni bir text dosyası yaratın ve adını “batsin.cpp” koyun. Daha sonra bu dosyayı bir editör ile (notepad, kate, emacs vs.) açın ve içine şu satırları yazın. Merak etmeyin birazdan bütün satırları açıklayacağız.

```
#include <iostream>

void main()
{
    std::cout<<“batsin bu dunya:)”<<std::endl;
    return;
}
```

Dosyayı kaydedin ve konsola geçin (genelde program yazarken bir editör penceresi ve bir konsol penceresini hep açık tutun) konsolda kaynak klasörüne gidin (ve hep orada kalın). Mesela Windows'da

```
> d:
> cd Kaynak
```

Linux'da ise:

```
>cd home/ahmet/kaynak
```

Şimdi kodu derlemek için gereken komutu verin. Mesela VS derleyicisi için

```
> cl batsin.cpp
```

gcc için ise:

```
> g++ batsin.cpp
```

Eğer dosyayı yazarken herhangi bir harf veya noktalama işareti hatası yapmadıysanız şimdi kaynak klasörünüzde yeni çalışabilir bir dosya olacak. Bu dosyanın adını konsola girdiğinizde de ilk programınız çalışacak ve o muhteşem mesajı göreceksiniz.

# Neler Oluyor

Şimdi programımızı inceleyeceğiz. Aslında bu en basit programda bile, iki satırda, oldukça ileri bazı konuların örnekleri var. Fakat aslında biz bu ileri konularla ilgili olan satırları çok basit bir amaç için kullanıyoruz. Kafanızın karışmaması için şöyle bir yol takip edeceğim. Şimdilik bu ileri düzeyde konularla ilgili olan iki satırı ne amaçla kullandığımızı söyleyip bir kalıp olarak ezberlemenizi isteyeceğim. Arka planda olan daha derin detayları daha sonra açıklamak üzere es geçeceğim.

Bahsettiğimiz bu iki satır birinci ve beşinci satırlar:

```
#include <iostream>

    std::cout<<“batsin bu dunya:)”<<std::endl;
```

Aslında büyük olasılıkla tahmin ettiğiniz üzere, bu satırları konsola programımızın yazı yazdırabilmesi için kullanıyoruz. Yazı yazdıran asıl komut beşinci satır. Tırnak işareti içindeki mesajı değiştirerek istediğiniz mesajı yazdırabilirsiniz. Bu satırdan birden fazla tekrarlayıp her birine farklı mesajlar da yazdırabilirsiniz. Mesela

```
#include <iostream>

void main()
{
    std::cout<<“batsin bu dunya:)”<<std::endl;
    std::cout<<“bitsin bu ruya”<<std::endl;
    return;
}
```

Birinci satırla ilgili bilmeniz gereken tek şey beşinci satırda gördüğünüz komutun çalışabilmesi için birinci satırın programınızın başında yer alması gerektiği.

Şimdi diğer satırları inceleyelim

```
void main()
{

    return;
}
```

Önce küme parantezlerine ( { ve } ) dikkatinizi çekmek istiyorum. Bu parantezler C++ dilinde kod satırlarını guruplamaya yararlar. Daha sonra göreceğimiz pekçok yapıda da bu gruplamayı görüyor olacaksınız. Bu örneğimizde küme parantezler programımızın tamamını grupluyor. Bu şekilde gruplanmış olan kod satırlarının oluşturduğu gruba “Kod Bloğu” denir.

Programın üçüncü satırında gördüğünüz int main() ifadesi burada kod bloğunun adını veriyor (aslında bu bir “fonksiyon” ve

“kod bloğu” ve fonksiyon tam olarak aynı şey değillerdir ama fonksiyonları daha ileride anlatacağız). “main” isminin özelliği C++'daki varsayılan program başlangıç noktasını tanımlamasıdır. Yani programınızda yüzbinlerce satır kod ve onbinlerce kod bloğu olabilir ama programınız çalışmaya “main” isimli kod bloğundan başlar.

Altıncı satırda kod bloğunun içinde gördüğünüz return komutu da kod bloğundan çıkmaya yarar. Bu örneğimizde sadece tek bir kod bloğu olduğu için de bu bloktan çıktığımızda programımız sona eriyor.

Üçüncü satırda geçen void kelimesinin ve parantezlerin ne anlama geldiğini birazdan anlatacağım. Şu an öğrenmenizi istediğim nokta aslında bir C++ programının temel iskeleti. Bu iskelete tekrar bakalım.

## Program İskeleti

Şimdi bir süre boyunca yazacağımız programları hep aynı iskelet içinde yazacağız. Bu iskeleti aşağıda görüyorsunuz.

```
#include <iostream>

void main()
{
    komut 1;
    komut 2;
    komut 3;
    ...
    ...
    komut n;
    return;
}
```

Bu iskelete sahip olan programlarımız n tane komutu arka arkaya calıştırıyor olacaklar. Dikkat ederseniz her komutun sonunda noktalı virgül işaretini kullanıyoruz. Bu işaret C++'da bir komutun bittiği anlamına gelir. Eğer noktalı virgülü unutursanız, derleyiciniz derleme esnasında hata verecektir. Programlarımız bu şekilde art arda komutları işlettikten sonra return komutunda belirtildiği şekilde kod bloğumuzu terkederek sonlanacaklar.

Şu ana kadar iki komut öğrendik. Birincisi şimdilik tam anlamadan ezberlediğimiz yazı yazdırma komutu, diğeri de return komutu. Daha başka komutlar öğrenmeye başlamadan önce değinmemiz gereken önemli bir konu var.

# Değişkenler

Değişkenler her programlama dilinde olan ortak bir kavramdır. Ben yıllar önce değişken kavramını ilk olarak C64 kullanma klavuzundaki BASIC öğreten bölümlerden öğrenmiştim. Hala kafamda oradaki cümlelerle bu konuyu ifade ediyorum. Şimdi size de aşağı yukarı oradakine paralel bir dil ile aktaracağım.

Değişkenleri içlerine bilgi koyup saklayabildiğiniz kutucuklar gibi düşünebilirsiniz. Bu kutulara her zaman yeni bir bilgi koyup her istediğimiz zaman da bu bilgiyi okuyabiliriz. Örneğin bir bilgisayar oyununda oyuncunun aldığı puanı saklayan bir değişken olabilir.

Her değişkenin programı yazan kişi tarafından tanımlanan bir “ismi” vardır. Örneğin oyuncunun aldığı puanları saklayan değişkenin ismini “skor” olarak belirleyebiliriz.

Değişkeni programda kullanırken amacımız, program çalışırken yapılan işlemlerin sonuçlarını saklamak ve lazım olduğunda sonra okumak, çeşitli olaylar sonucu değişkenin sakladığı değeri değiştirmek gibi çeşitli uygulamalardır. Örneğin Skor değişkenimizi programda tanımladıktan sonra, aşağıdaki sıralanan şekillerde kullanabiliriz.

1. Oyuncu yeni oyuna başlarken skora sıfır yaz.
2. Oyuncu bir düşmanı vurunca skora 5 ekle.
3. Ekrana oyuncu skorunu yazarken şu anki skoru skor değişkeninden oku.

## Değişkenlerin Tipleri

C++'da her değişkenin bir “tipi” vardır. Bir değişken sadece kendi tipinden bir bilgiyi saklayabilir. Örneğin bir değişkenin tipi “tam sayılar” ise bu değişkene “yazı” tipinden bir bilgi yazamayız. Tip, bir değişkenin üzerinde yapılabilecek işlemleri tanımlar. Örneğin tamsayı tipindeki bir değişkeni aynı tipteki başka değişken ve sabitlerle “toplama” işlemine sokabiliriz. Ancak bir tamsayı ile bir yazıyı toplayamayız (Elma ile armutu toplamayın derdi ya ilkokul öğretmenlerimiz onun gibi :) )

C++ dilinde en sık kullandığımız temel tipler arasında şunlar vardır:

1. char. Yani tek harf veya sembol. Örneğin a, K, z, :, ?, +
2. int. Yani tamsayılar. Örneğin 12, -185, 12456
3. float. Yani ondalıklı sayılar. Örneğin 2.57, -0.568, 458.366

## Değişkenlerin İlan Edilişi

Programınızda bir değişkeni kullanabilmeniz için bu değişkeni ilan etmeniz (declaration) gerekir. Bir değişkenin ilan edilmesi aşağıdaki gibidir

```
Tip isim;
```

Örneğin

```
int skor;
```

Burada skor adında ve tamsayı tipinde bir değişken kullanacağımızı ilan etmiş oluyoruz. Bunun sonucu olarak bilgisayar şu demin bahsettiğimiz gibi bir “kutucuk” hazırlıyor bellekte, ve bu kutucuğa skor ismini veriyor. Aynı zamanda kutucuğun saklayabileceği değerlerin tamsayı tipinde olması gerektiğini de saklıyor. Biz eğer programımızın başka bir yerinde yanlışlıkla skor değişkenine tamsayı tipinde olmayan bir değer yazmaya çalışırsak derleyici bize hata döndürecek.

Bu şekilde değişkenimizi tanımladıktan sonra içine yazıp okuyabiliriz. Örneğin aşağıdaki programa bir bakın

```
#include <iostream>

void main()
{
    int skor;
    skor = 0;
    std::cout<<skor<<std::endl
    return;
}
```

Burada ilk komut ile skor değişkenini ve onun tipini ilan ettik. Ardından gelen komut ile bu değişkenin içine 0 değerini yazdık. Ardından sihirli “konsola yazdırma” komutumuz ile skor değişkeninden okuduğumuz değeri ekrana yazdırdık.

# Gelecek bölümde

Kursun bu ilk bölümünü burada kesiyoruz. Anlamadığınız yerler veya gördüğünüz hatalar olması durumunda bize yazmaktan çekinmeyin. Gelecek bölümde kullanıcıdan veri okuma, sabitlerin kullanımı, ve if komutu ile tanışacağız. If komutu, en önemli komutlardan biri ve şu ana kadar yazdıklarımızdan çok daha ilginç programlar yazabilmemizi sağlayacak.

# Gnu Make ve Makefile Kullanımı

Emirhan (Ragnor) Bayyurt

Make Unix türevi işletim sistemlerinde sıkça kullanılan popüler bir otomatik yazılım derleme aracıdır. Makefile adı verilen dosyalardaki ayarları kullanarak tek komutla projelerin derlenmesini sağlar. Bu dosyalarda sizin ayarlayacağınız parametrelerle projenizin derlenmesi, kurulması, sistemden kaldırılması ya da derlenmiş dosyaların silinmesi gibi işlemler tek komut ile kolayca halledilebilecek hale gelir.

En basit kullanım şekli sadece 'make' komutunun konsolda yazılmasıyladır. Sıklıkla kullanılan diğer parametreler, 'make install', 'make clean' dir. 'make' komutu projeyi derler, 'make install' derlenen projenin sisteme kurulmasına ve 'make clean' de derlenmiş olan proje dosyalarının silinmesine yarar. Siz bu parametre isimlerini kullanmak zorunda değilsiniz ya da parametrelerinizi bunlarla sınırlamak durumunda da değilsiniz ama belli başlı görevler için (kurulum, silme vs.) en fazla kullanılan parametreler bunlardır.

Konsolda 'make' komutunu verdiğinizde make programı o an içinde bulunduğunuz klasörde bir makefile dosyası arar. Bu dosyanın sadece adı 'Makefile' ya da 'makefile' olabilir. Başka bir isme sahip bir makefile dosyası kullanmak istiyorsanız bunu -f parametresi ile make programına bildirmeniz gerekir.

```
make -f makefile.win
```

Üstteki örnekte make programı 'makefile' ya da 'Makefile' adında bir dosya aramak yerine sadece 'makefile.win' adında bir dosya arayacaktır.

Bir makefile dosyasının yapısı şu şekildedir.

```
hedef: gereklilikler
[tab] komutlar
```

Oldukça basit bir makefile örneği de şöyle olabilir.

```
all: proje_adi

proje_adi: main.c
	gcc -o projem main.c

clean:
	rm -rf *.o
	rm -rf projem
```

Herhangi bir parametre olmadan 'make' komutu verildiğinde çalıştırılan parametre 'all' parametresi olur. 'all' parametresi 'proje_adi' parametresini çalıştırır. O parametrede de main.c dosyası derlenerek projem dosyası oluşturulur. Eğer 'make clean' komutu verilirse klasörde bulunan bütün .o uzantılı dosyalar ve projem dosyası silinir. '.o' uzantılı dosyalar gcc tarafından oluşturulan obje dosyalarıdır. Make'in bir avantajı da bu dosyalarda bir değişiklik yapılmamışsa makefile'da o dosyaya ait olan derleme komutlarını atlar. Bu sayede tekrar tekrar derlemeler yaptığınız projelerinizde derleme zamanları oldukça kısalır.

Makefile dosyalarında sık sık tekrar ettiğimiz komutları bir değişkene atayıp o değişkeni çağırma şansımız vardır. Bu özellik sıklıklar derleyici adı, kütüphane dosyaları, proje dosyaları gibi tekrar eden metinler için kullanılır.

```
LIBS=-L"/usr/lib" -lSDL -lGL
```

Yukarıdaki şekilde tanımladığımız değişkeni makefile içinde kullanmanın yolu ise $(LIBS) şeklinde yazmaktır.

Örnek olarak şöyle bir makefile yazabiliriz.

```
CC=gcc

LIBS=-L"/usr/lib" -lSDL
EXE=projem

all: proje

proje: main.c
	$(CC) -o $(EXE) main.c $(LIBS)
```

Üstte yazdığımız örnek çok basit kaçmış olabilir ama sadece bir kaynak dosyası yerine 5 ya da daha fazla kaynak dosyanız olduğunu, projede 4-5 farklı kütüphane de kullandığınızı hayal edin. İşte o zaman bu özellik işleri oldukça kolaylaştırıyor.

Bunların dışında makefile'ınız içinde açıklama satırı yazmak için satırın başına '#' karakterini koymanız yeterlidir. Uzun satırları bölmek için '\' karakterini kullanıp, alt satırdan yazmaya devam edebilirsiniz. $@ değişkeni o anda kullanılan hedefin adını belirtir, aynı şekilde $< değişkeni de kullanıldığı andaki girdi dosyasını belirtir. Son olarak dikkat etmeniz gereken şey ise en başta makefile yapısını gösterir verdiğim örnekte de yazdığı gibi hedef'in altındaki komutları yazarken başına boşluk bırakmayacaksınız, orada kullanmanız gereken şey 'tab' karakteridir. Boşluk bırakırsanız make programı hata verip sonlanacaktır.

Son olarak size projelerinizde kullanmanız için ufak ama etkili bir makefile örneği verip yazımı burada sonlandırıyorum.

```
# projenizde kullanacaginiz derleyiciler
CPP  = g++
CC   = gcc

OBJ=$(SOURCES:.cpp=.o)

# projenizde kullandiginiz kutuphaneler
# bu ornekte ben sdl kitapliklari kullaniyorum
LIBS =  -L"/usr/lib" -lSDLmain -lSDL \
              -lSDL_mixer -lSDL_ttf

# projede kullandiginiz kutuphanelerin .h
# dosyalarinin bulundugu klasorler
INCS =  -I"/usr/include" -I"/usr/include/SDL"
CXXINCS =  -I"/usr/include"
CXXFLAGS = $(CXXINCS)
CFLAGS = -c $(INCS)

# sources degiskenine projenizdeki
# kaynak kod dosyalarini (.c yada .cpp)
# aralarinda bir bosluk birakarak yazin,
# en basta main dosyaniz olsun
SOURCES= main.cpp

# buraya projeniz derlendiginde cikacak
# executable dosyanin adi yazilacak
BIN  = proje_adi
RM = rm -f

all: $(SOURCES) $(BIN)

$(BIN): $(OBJ)
    $(CC) $(OBJ) -o $(BIN) $(LIBS)

.cpp.o:
    $(CC) $(CFLAGS) $< -o $@

clean:
    ${RM} $(OBJ) $(BIN)
```

Hepsi bu kadar. Umarım yazım size faydalı olmuştur.

# Bilgisayar Mimarisine Giriş 1

Bilgem (Nightlord) Çakır

Bu yazıda her programcının bilmesi gerektiğine inandığım bazı temel kavramlardan bahsedeceğim. Bu kavramlar programcılık üzerine çalışmak isteyen ve yeni başlayan kişilerin daha hızlı öğrenmelerini sağlarken, uzun süredir programcılık yapan fakat elektronik tabanı fazla olmayan kişilerin de bilgi dağarcıklarında kalan olası boşlukları doldurmakta yardımcı olabilir.Kısa kısa değineceğim konu başlıkları şunlar:

1. Register(yazmaç) nedir?
2. CPU nelerden olusur?
3. Bellek nedir?
4. Bir CPU bellekteki programı nasıl çalıştırır?
5. Komutlar (instruction) neye benzer?
6. Makine kodu komutlardan Assembler'a geçiş

## Register Nedir

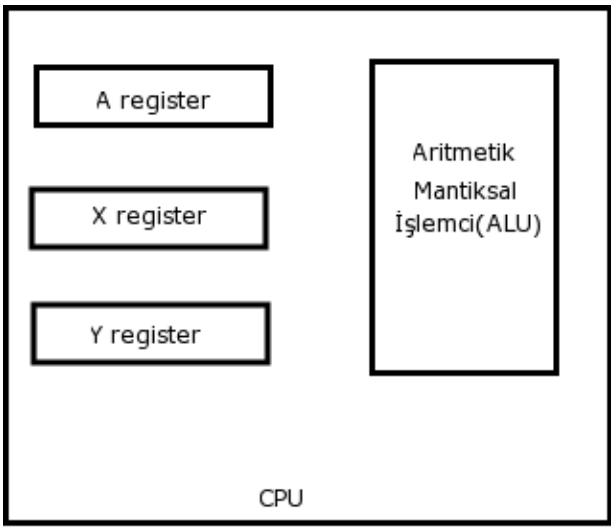
Bilgisayar mimarisinden bahsederken önce registerları incelemek lazım. Dijital elektronikte en küçük bilgi saklayabilen birim flip flop adı verilen devredir. Bu devreye 1 veya 0 değerlerinden birini verirseniz, siz başka birşey yapana kadar bu değeri saklamayı becerir. Böylece bir bitlik bir bilgiyi saklamış olursunuz. Eğer 2 flip flopunuz varsa bu ikisine 1'ler ve 0'lar saklayarak toplam 4 farklı değeri taşıyabilen bir hafıza devresi elde edebilirsiniz. İşte registerlar bu şekilde 4 veya 8 veya başka sayıda flip flopların yan yana getirilmesiyle oluşturulur. Biz genelde 8 bitlik (yani 8 flip flopluk) registerlarla uğraşacagız.

8 bitlik bir registerdaki her bit'e 1 veya 0 yazarak toplam 256 değişik değer yazabiliyoruz. Bu değerler genelde ikilik sayı düzeninde 8 basamaklı bir sayı olarak ifade edilir. 10010111 veya 01101111 gibi... Bunları genelde onluk düzendeki 0 dan 255'e kadar olan sayılar olarak yorumluyoruz. Bazen de -128'den +127'ye kadar ki sayılar olarak yorumlayabiliriz. Programdaki amacımıza göre bu bit dizilerine farklı anlamlar yükleyebiliriz. Ama her koşulda toplam 256 değişik değerden fazlasını yazamıyoruz, bunu unutmayalım. Böyle 256 değişik değer alabilen bir bilgi kutucuğuna bayt (byte) adı verilir. Yani 64 kilo byte'lık bir dosya yaklaşık 64000 tane 0 ile 255 arası sayının oluşturduğu bir bilgi grubudur.

Yeniden bir toparlayalım. Register nedir? Register içine 0'dan 255'e kadar bir sayı yazılabilen ve bu değeri unutmayan kutucuklardır. Şimdi CPU yaparken registerların oynadığı rolü inceleyelim. (Not: registerların tek görevi bilgi saklamak değildir aslında ama şimdilik konuyu karıştırmayalım.)

## CPU(işlemci)

Bu noktada bir CPU tasarımı yaptığımızı farzedelim. İşleri basit tutmak için ilk etapta 8-bitlik (bunun tam olarak ne demek olduğunu zonra anlayacaksınız) bir CPU tasarlayacağız. Bu basit CPU'muza 3 tane 8 bitlik register koyalım ve bunlara A, X, Y adlarını verelim. Bu CPU herhangi bir anda içinde 3 tane 8-bitlik deger (yani 3 bayt) saklayabiliyor. Gerçekte bir CPU'da bundan fazlası var ama şimdilik devam edelim.




**Şekil 1.**

Dikkat ederseniz su ana kadar registerlarla yapabildiğimiz tek işlem içlerine bilgi saklamak. Fakat bir CPU yapmaya kalkıştığınızda CPU'nuzun bir takım bilgi işleme özellikleri de olmalıdır. Mesela toplama, çıkarma gibi aritmetik işlemler ya da VE / VEYA gibi mantıksal işlemler yapabilmemiz için registerlardan fazlasına ihtiyaç var. İşte bu noktada yardımımıza ALU koşuyor. ALU birtakım aritmetik işlemleri yapabilen kombinasyonel bir devredir.

## Kombinasyonel devre nedir?

Dijital elektronikte bütün devreler iki gruptan birisine mensuptur. Kombinasyonel devreler ve durumlu(state) devreler. Kombinasyonel devreler, içlerinde herhangi bir durum bilgisi taşımayan devrelerdir. Bu devrelerin girişleri ve çıkışları arasında bir bağıntı olur. Giriş değerlerinin herhangi bir andaki değerini bilirseniz çıkış değerini de bilirsiniz. Mesela mantık kapıları bu sınıfa girer.

Örneğin 2 girişli bir VE kapısını ele alalım. Girişlerden her ikisi de 1 olursa çıkış 1'dir. En az bir giriş 0 ise çıkış 0'dır. Dolayısıyla biz iki girişe 1 veya 0 yazdığımızda sonucun ne olacağını biliriz. Girişlerin değerlerinin daha önce ne olduğunun önemi yoktur. Başka bir deyişle, devrenin bir durumu ya da hafızası yoktur.

Oysa flip floplarda durum tam tersidir. Örneğin bir flip flop türü olan toggle- flip flop lar şöyle çalışır. Bu flip flopun bir girişi vardır. Bu girişe toggle adı verilir. Çıkış ise flip flopun o an sakladığı bilgidir (bir bitlik bilgi yukarıdan hatırlayın). Toggle değeri 0 iken değer değişmez. Toggle 1 yapılırsa çıkış değişir. 0 ise 1, 1 ise 0 olur. Şimdi dikkat ederseniz bu devrenin giriş değerini bilmemiz çıkışı tahmin etmemize yetmiyor. Devre aynı girişe farklı durumlarda farklı çıkış veriyor. Yani devrenin bir durumu başka bir deyişle bir hafızası var.

Şimdi bu temel kavramı toparlayalım. En basitten en karmaşığa kadar bütün dijital devreler bu iki tip alt devrelerin birleşiminden oluşur. Bu iki tür devre tipine sık sık değineceğim.

# ALU(Arithmetic Logic Unit)

Şimdi ALU konusuna dönelim. ALU nun kombinasyonel (yani hafızası olmayan) bir devre olduğunu söylemiştim. ALU giriş olarak iki tane 8 bitlik hat ve yapılacak işlemi seçen bazı kontrol hatları alır. Çıkışı da 8 bitlik bir hattır. CPU içinde genelde registerlar ALU'nun girişlerine dijital anahtarlarla (ki bunlara bus transceiver da denir) bağlanır. Bu anahtarlar kapatılırsa o registerda saklanan değer ALU ya giriş parametresi olarak verilmiş olur. Ayrıca bellekten okunan bir değer de ALU girişine bağlanabilir. Bu bağlantı yapılıp kontrol girişlerine de yapılması istenen operasyon belirtildiği anda ALUnun çıkışında sonuç belirir. Bu cıkış yine dijital anahtarlarla bir registera bağlanabilir. Böylece 2 registerla veya bellekle yapılan bir işlemin sonucu tekrar bir registera saklanmış olur.

Bu bölümü sağ salim atlatabildiyseniz ciddi bir tebriği hakettiniz. Şu anda CPU mimarisi ile ilgili çok temel bazı kavramları anlamış bulunuyorsunuz. Eğer anlamadıysanız çok önemli değil ama arasıra tekrar okuyarak ya da bize yazarak bu problemi çözmeye çalışın.

Özet olarak CPUmuzda registerlar ve ALU sayesinde artık registerlara bir takım değerler yüklüyor bunlar arasında işlemler yapıp sonuclar tekrar registerlara saklayabiliyoruz. Bu CPU ile şu halde pekçok faydalı iş yapılabilir. Fakat biz bilgisayar dediğmiz alete doğru basitten karmaşığa gelişen bir yolculuğa devam edeceğiz. Önümüzdeki konu bellek

# Bellek

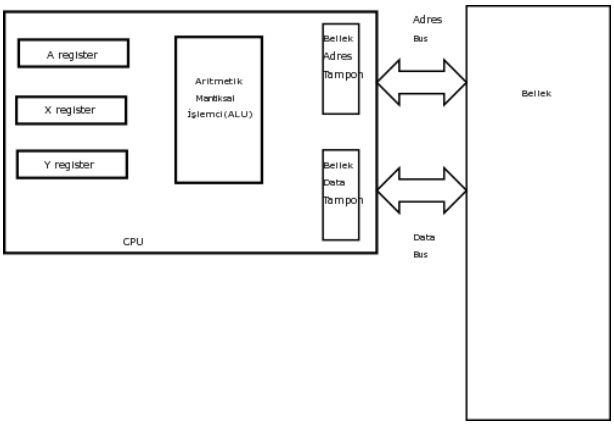
Bellek her biri registerlara benzayen 8'er bitlik saklama hücrelerinin bir araya getirilmesiyle oluşturulan hafıza bloğudur. Registerlardan daha ucuz ve daha yavaştır. Bu yüzden daha bol miktarda konulabilir.

Bellekteki her bayta erişebilmimizi sağlayan şey adresleme kavramıdır. Örneğin 64 kilobaytlık bir bellekte adresleri 0'dan başlayıp yaklaşık 64000 civarlarına kadar uzanan yaklaşık 64000 kutucuk vardır. Bu kutucukların her biri bir baytlık (0-255 arası) bir değer alabilir.

Bellek çipleri üzerinde adres ve data bus adı verilen pinler olur. Bir pin ile de okuma veya yazma seceneği seçilir. Mesela okuma/yazma pinine okuma seçeneğini koyup adres bus'a 15 yazarsak, 15. adresteki baytın değeri data busda gözükür.

Bus nedir? Dijital elektronikte çipleri veya çiplerin içindeki alt bölümlerin birbirleriyle data alışverişi yapmalarını sağlayan çoğu zaman 8, 16, 32 bitlik hatlara bus denir. Mesela CPUdan bahsederken ALUnun çıkışının A registerine bağlı olduğunu söylemiştim. İşte bu bağlantı ALU devresinin çıkışındaki 8 bağlantı noktasından A registerinin girişindeki 8 giriş bağlantı noktasına giden yan yana 8 kablodan ibarettir.

İşte CPU ve belleği bir araya koyarsak daha çok iş yapabilen bir CPU elde edebiliriz. CPU genelde Bellek Adres Tamponu ve Bellek Data Tamponu denilen registerlar içerir. Bu registerlar bellek çipinin adres ve data buslarına bağlıdır. Böylece CPU Adres Tamponuna bir değer yazıp, bellekte o adreste bulunan data değerini Bellek Data Tamponuna yükleyebilir. Böylece o registerdaki değer herhangi bir diğer registera kopyalanabilir. Bu register da ALU ya bağlıdır. Böylece bellekten okunan bir değer o an registerlarda bulunan değerlerle aritmetik/mantıksal bi işleme sokulabilir. Son olarak registerlardan birindeki değer Bellek Data Tamponuna kopyalanıp ardından Bellek Adres Tamponuna istenen bir adres yazılır ve Okuma yazma pini de Yazma durumuna getirilirse, CPU bellekteki istediği bir adrese herhangi bir registerın değerini yazmış olur.




**Şekil 2.**

Şu an önemli bir noktaya geldik. Artık elimizde bayağı yetenekli bir cihaz var. Elimizdeki bellekte duran bilgileri okuyup registerlara yüklüyor ardından üzerlerinde bazı işlemler yapıyor ve sonuçları tekrar bellekte saklayabiliyoruz. Bir örnek inceleyelim... Diyelim ki bellekteki 10 adresindeki sayı ile 15 adresindeki sayılar toplanıp sonuç 100 adresine yazılacak. olaylar şu sırayla olur.

1. Adres Tamponuna 10 yazılır. Okuma modu secilir. Data tamponuna 10 adresindeki data gelir.
2. Data tamponundaki değer A registerine kopyalanır.
3. Adres Tamponuna 15 yazılır. Okuma modu secilir. Data tamponuna 15 adresindeki data gelir.
4. ALUda toplama işlemi seçilir. ALUnun bir inputunda A diğer inputunda Data tamponu bağlantıları açılır. Toplamanın sonucu A registerine geri yüklenir

5. A registerindeki değer Data Tamponuna kopyalanır.
6. Adres Tamponuna 100 yazılır. Yazma modu seçilir. Değer belleğe yazılmış olur.

# Komutlar (Instructions)

Şu ana kadar bellekten bahsederken hep data saklamak amacı ile bahsettik. Ama bellekte saklanabilen bir diğer veri de komutlardır. Komutlar da aslında sayısal değerlerdir diye düşünebilirsiniz. Sadece onları komut gibi yorumladığımızı düşünelim. Mesela 20 sayısını topla komutu diye yorumlayabiliriz. Komutlar çoğu zaman bir komut ve arkasından gelen komutla ilgili argumanlar olarak gelirler. Mesela "(A registerini yükle)(30)" gibi bir komut düşünülebilir. Eğer bu komutun numarası diyelim ki 25 ise biz belleğe arka arkaya 25 ve 30 sayılarını saklarsak bu komutu belleğe saklamış oluruz.

## Program sayacı (Program Counter)

Bellekte ardarda duran komutların işletilmesi işini çözmek üzere CPUmuza bir register daha ekliyoruz. Bu register içinde saklanan sayı bellekten okunacak olan komutun adresini gösteriyor. Bellekten her komut okuyuşumuzda, komutu işledikten sonra program sayacı bellekte bir sonraki komutun olduğu yeri gösteriyor.

## Kontrol Devresi

Şimdi kafanız karışmış olabilir. Program sayacının değerini kim artırıyor? Ya da bellek erişimlerinde Adres ve Data Tamponlarına kim değerleri koyuyor. Ya da ALU ya hangi işlemin yapılacağını kim söylüyor. İşte bütün bu işleri CPU içinde yöneten bir bölüm var ki buna Kontrol Devresi deniyor(Control Logic). Bu devre Bellekte Program sayacının gösterdiği yerdeki komuta göre registerların arasındaki dijital anahtarları açıp kapatarak CPU içinde datanın nereden nereye gideceğini yönlendirmekle kalmaz, ALUnun yönetimini, belleğe giden okuma/yazma pininin durumunu vs kontrol eder. Bir komutu tamamladıktan sonra Program sayacını değiştirip bir sonraki komuta bakmasını sağlar. Bir CPUnun mimarisindeki en önemli ögelerden biridir.

## Jump komutu

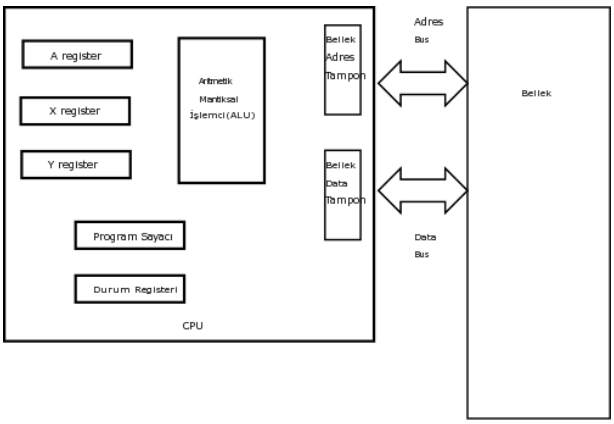
Bu komut ile Program sayacı bellekteki bambaşka bir noktaya bakacak şekilde set edilerek program akışının oradan devam etmesi sağlanabilir.

# Durum Registeri

Şimdi CPUmuza bir register daha ekliyoruz. ALU ile yaptığımız işlemlerin sonucunda ortaya çıkan durumlarla ilgili bazı faydalı bilgileri kontrol devresi buraya da yansıtır. Bu iş şöyle yapılır. Bu registerdaki bitlere özel anlamlar yüklenir. Örneğin bir tanesine Zero-flag denir. Bu bit ALUdaki işlemin sonucu sıfır çıkarsa 1 olur. Bir diğer bite Carry-flag denir. Bu da ALUdaki toplama işleminde sonuç 255'i geçerse 1 olur. Bir diğer bitte Negative-flag'dir. Bu da ALUdaki işlemin sonucu negatif çıkarsa (mesela 2 - 30) 1 olur.

Bu bayraklar bizim için çok önemlidir. Çünkü bu bayrakların durumuna bakarak hareket eden komutlar vardır. Bunlara Koşullu Dallanma (Conditional Branching) komutları denir. Bu komutlar sayesinde programlara karar mekanizmaları eklenebilir. Örneğin şöyle kodlar yazılabilir:

1. bellekte N adresindeki değeri A'ya yükle
2. A daki değer ile 30'u karşılaştır (ALU ile A'dan 30'u çıkar)
3. Sonuç 0 ise 6. basamağa atla (işte burası koşullu dallanma komutu. Eğer koşul sağlanmazsa program bir sonraki komuttan devam edecek)
4. N adresindeki değer 30 olmadığında neler yapacaksan onu yap
5. Programdan çık
6. N adresindeki değer 30 olduğu zaman neler yapacaksan onu yap.
7. Programdan çık




**Şekil 3.**

# Komutlara özet bakış

Şu ana kadar incelediğimiz makine dili komutları dikkat ederseniz sadece aşağıdakilerden birini yapabilir.

1. Bellekten bir registera değer kopyalama
2. Bir registerdan belleğe değer kopyalama
3. Jump ile bellekte herhangi bir yerdeki bir komuta atlama
4. İki register arasında değer kopyalama

5. Register ve bellek arasında aritmetik/mantıksal bir işlem yapıp sonucu tekrar bir registera kopyalama

6. ALUdaki işlem sonucuna göre dallanma veya bir sonraki komuta devam etme.

Gerçek CPU'larda da komutların %75'i bu saydıklarımızdan birini veya birkaçını yapar ama bunların dışında birşey yapmaz. Yani herbiri küçük ama faydalı işler yapar. En karmaşık programlar ile bu küçük işlerin bir araya gelmesiyle oluşur. Bu yüzden Makine dili komutlarını öğrenmek kolay bir iştir. Makine dili ile program yazmak ise problemleri en küçük parçalarına kadar bölerek tasarım yapmayı öğrenene kadar size zor gelebilir.

# Makine Dili Komutlarından Assembler'a geçiş

İnsanlar ilk yaptıkları bilgisayarları programlamaya çalışırken bellekteki her değeri rakam rakam hazırlarlarmış. Hatta rivayetlere göre en eski bilgisayarlardan olan ENIAC da bu rakamları belleğe girerken binlerce elektrik anahtar (yani bildiğimiz ampul yakan cinsten) çat çut teknisyenlerce açılır kapanırmış. Tabii fazla geçmeden insanlar program yazarken ince ince sayılarla uğraşmasın diye makine dili komutlarını genelde 3 harfli kısaltmalarla ifade eden assemblar dillerini oluşturmuşlar. Böylece mesela 24,125,34,55,98,12,90 diye program yazmaya çalışmak yerine insanlar

```
lda 24  (yani A registerine 24
         adresindeki degeri yukle)
clc     (yani Carry Flag'i sifirla)
adc 98  (98 adresindeki deger ile
         Carry Flag degerini toplayip
         A registerine ekle)
sta 90  (sonucu 90 adresine yolla)
```

diye kod yazabilmeye başlamış. Daha sonra assemblerlar daha da gelişmiş ve insanlar çeşitli rakamların yerine isimler (bunlara etiketler de diyoruz) kullanabilmeye başlamış. Böylece yukarıdaki kod yerine

```
mevcut_param = 24
maas = 98
ev_sahibi = 90

lda mevcut_param
clc
adc maas
sta ev_sahibi
```

yazabilmeye başlamışlar.

İşte günümüzde Makine dili kullanarak kod yazan hemen herkes bu tip assembler araçları kullanır. Bu araçları kullanırken yazılan kod nasıl düzenli tutulur? Aradan 5 ay geçtikten sonra eski kodlarınıza döndüğünüzde onlardan birşeyler anlayabileceğinizi nasıl garanti edersiniz? Bütün bunlar cevaplanması gereken önemli sorulardır. Bunlarla ilgili yazılım mühendisliğinde kullanılan bir ton teknik vardır. Bunlar da başka bir yazının konusu.

"Eee biz şimdi hiçbir assembler komutu öğrenmedik" diyebilirsiniz. Panik yapmayın onun da zamanı gelecek. Hali hazırda bu konuda bulabileceğiniz kaynaklar var internette. Şimdi gidin ve onlara bir daha bakın. Pek çok şeyi daha kafanızda pekiştirerek anladığınızı göreceksiniz. Ayrıca ben de er geç o konuda birşeyler yazacağım.

# Sonuç

Bu yazı aslında giriş seviyesinde bir yazı olmakla beraber içerdiği bazı elektronik detayları yüzünden orta seviye gibi gelebilir. Eğer yazının içinde anlamadığınız yerler varsa bize yazarak sorabilirsiniz. Sonuçta anlamadığınız yerler kalsa bile anladığınız yerler size bayağı fayda sağlayacaktır. Ayrıca sizinle bire bir çalışan bir hocanız olmadıkça her konuda yeni birşeyler öğrenirken, bazı yerleri anlamayabilirsiniz. Daha sonra başka bir yerde birşey okurken anlamadığınız şeyle alakalı olduğunu anlar ve gereken ilişkilendirmeyi kafanızda yaparsınız. Zaten öğrenme böyle bir deneyimdir.

Önümüzdeki sayıda Giriş/Çıkış sistemlerinin CPU ve bellek ile nasıl entegre edildigini inceleyeceğiz.

# PHP : Hypertext Preprocessor - Bölüm 2 - Plazma

Emir (Skate) Akaydın

## Giriş

Merhaba sevgili Plazma okurları.

Geçen bölümde PHP diline bir giriş yapmıştık. PHP'nin sunucu tarafında çalışan bir dil olmasına değinmiş ve PHP dilini bazı istemci tarafında çalışan dillerle mukayese etmiştik. İlk program örneklerinin ardından "Değişkenler" ve "Fonksiyonlar" gibi iki önemli başlığı işlemiş ve bu noktada yazımıza noktayı koymuştuk.

PHP yazı dizisinin ikinci bölümü olan bu bölümde sınıflar konusunu işleyeceğiz.

**Şekil 1.**

## PHP Dilinde Sınıf (Class) Kullanımı

PHP 4 versiyonunda örnek bir sınıf aşağıdaki gibi tanımlanmakta ve kullanılmaktadır.

PHP 4'de Film sınıfı:

```
<?php
class Film
{
    var $isim;
    var $fragman;
    var $sureDakika;

    function Film($filmIsim = "",
                  $filmFragman = "",
                  $filmSureDakika = 0)
    {
        $this->isim = $filmIsim;
        $this->fragman = $filmFragman;
        $this->sureDakika = $filmSureDakika;
    }

    function uzunluk()
    {
        if($this->sureDakika < 30)
            return "Kisa film";
        else if($this->sureDakika < 90)
            return "Normal sureli film";
        else
            return "Uzun film";
    }
}

$jackass2 = new Film("Jackass II", "jtm2.avi", 95);

echo "film ismi: ".$jackass2->isim."<br>";
echo "film suresi: ".$jackass2->sureDakika." dk.
      <br>";
echo "film uzunlugu: ".$jackass2->uzunluk();
?>
```

PHP 5 ile birlikte sınıflara birçok yeni özellik eklendi. Artık yukarıdaki örnek şu şekilde tanımlanmaktadır.

PHP 5'de Film sınıfı:

```
<?php
class Film
{
    private $isim;
    private $sureDakika;
    private $fragman;

    public function __construct($filmIsim = "",
                                $filmFragman = "",
                                $filmSureDakika = 0)
    {
        $this->isim = $filmIsim;
        $this->fragman = $filmFragman;
        $this->sureDakika = $filmSureDakika;
    }

    public function uzunluk()
    {
        if($this->sureDakika < 30)
            return "Kisa film";
        else if($this->sureDakika < 90)
            return "Normal sureli film";
        else
            return "Uzun film";
    }

    public function okuIsim()
    {
        return $this->isim;
    }

    public function okuSureDakika()
    {
        return $this->sureDakika;
    }
}

$jackass2 = new Film("Jackass II", "jtm2.avi", 95);

echo "film ismi: ".$jackass2->okuIsim()."<br>";
echo "film suresi: ".$jackass2->okuSureDakika().
     " dk.<br>";
echo "film uzunlugu: ".$jackass2->uzunluk();
?>
```

PHP 4 ve PHP 5 arasındaki temel farklar şunlardır.

- PHP 4'de sınıfların constructorları sınıf ile aynı ismi taşıyan fonksiyonlar ile tanımlanırlar. PHP 5'de ise bu iş için "__construct" isimli sabit fonksiyonlar kullanılmaktadır.

- PHP 5'de public, private, protected gibi C++ standartlarına uygun tanımlar eklenmiştir. PHP 4'de ise bir sınıfın içersinde yer alan herşey public kabul edilmektedir.
- PHP 4'de olmayan, C++ standartlarına uygun birçok diğer özellik de yine PHP 5'de eklenmiştir.

Şimdi PHP 5'deki sınıfların başlıca özelliklerini bir bir inceleyelim.

# $this Kelimesi

Sınıfların içersinde tanımlanan değişkenlere sınıfların metotlarından (metot: sınıfa ait fonksiyon) ulaşmak istenildiğinde $this kelimesini kullanırız. Örnek:

```
class Film
{
    private $isim;

    public function okuIsim()
    {
        return $this->isim;
    }
}
```

# Türetilmiş Sınıflar

Birbirleriyle birçok ortak özelliği olan sınıflar birbirlerinden türetilebilirler. Örnek olarak Sürüngen ve Kertenkele sınıflarını ele alalım. Sürüngen sınıfı herhangi bir sürüngenin yaş, boy ve sürünme hızı bilgilerini tutsun. Bu bilgiler Kertenkeleler için de geçerli olacaktır. Ancak Kertenkelelerde her süründen de bulunmayan kuyruk mevcuttur. Buna bağlı olarak Kertenkele sınıfına bir de kuyruk uzunluğu kriteri eklemek isteyebiliriz. Yapmamız gereken öncelikle Sürüngen sınıfını tanımlayıp, daha sonra Sürüngen sınıfından türetilen Kertenkele sınıfını oluşturmaktır.

Bir sınıftan yeni bir sınıf türetmek için "extends" kelimesi kullanılır. Örnek:

```
<?php
class Surungen
{
    // Degisken Tanimlari
    private $yas;
    private $boy;
    private $surunmeHizi;

    // Metot Tanimlari
    public function okuYas()
    {
        return $this->yas;
    }


    public function yazYas($y)
    {
        $this->yas = $y;
    }

    public function okuBoy()
    {
        return $this->boy;
    }

    public function yazBoy($b)
    {
        $this->boy = $b;
    }

    public function okuSurunmeHizi()
    {
        return $this->surunmeHizi;
    }


    public function yazSurunmeHizi($sh)
    {
        $this->surunmeHizi = $sh;
    }
}

class Kertenkele extends Surungen
{
    // Degisken Tanimlari
    private $kuyrukUzunlugu;

    // Metot Tanimlari
    public function okuKuyrukUzunlugu()
    {
        return $this->kuyrukUzunlugu;
    }


    public function yazKuyrukUzunlugu($ku)
    {
        $this->kuyrukUzunlugu = $ku;
    }
}

// Kertenkele sinifindan kertenkelemizi tanimlayalim
$kertenkelem = new Kertenkele();

// Kertenkelemizin ozelliklerini tanimlayalim
$kertenkelem->yazYas(2);
$kertenkelem->yazBoy(15);
$kertenkelem->yazSurunmeHizi(7);
$kertenkelem->yazKuyrukUzunlugu(6);

// Kertenkelemizin ozelliklerini ekrana yazdiralim
echo "Yas: ".$kertenkelem->okuYas()."<br>";
echo "Boy: ".$kertenkelem->okuBoy()."<br>";
echo "Surunme Hizi: ".$kertenkelem->okuSurunmeHizi().
     "<br>";
echo "Kuyruk Uzunlugu: ".
     $kertenkelem->okuKuyrukUzunlugu()."<br>";
?>
```

Bu örnekte dikkat etmemiz gereken özellik şudur. "$kertenkelem" ismiyle tanımladığımız nesne, "Kertenkele" sınıfından oluşturulmuş bir nesnedir. Bu nesnenin yaş, boy, sürünme hızı özelliklerini kullanıyoruz ancak "Kertenkele" sınıfında böyle özellikler tanımlı değil. Bu özellikler sınıfı tanımlarken kullandığımız türetme yönteminden geliyor. "class Kertenkele extends Surungen" dediğimiz için Kertenkele sınıfı Surungen sınıfının tüm özelliklerine sahip bir şekilde yaratılıyor.

# Yapıcı ve Yıkıcılar

Her sınıf oluşturulurken ve yok edilirken çağırılan fonksiyonlar vardır. Bunlara "yapıcı" ve "yıkıcı" (constructor / destructor) fonksiyonlar denir. PHP'de yapıcı fonksiyon "function __construct()" şeklinde tanımlanmıştır ve gerektiği taktirde parametre alabilir. Yıkıcı fonksiyon ise "function __destruct()" şeklinde tanımlanmıştır. Parametresiz kullanılması gerekmektedir.

PHP'de yıkıcı fonksiyonlar çok sık kullanılmazlar. Çünkü genel-

likle PHP'de bir scriptin çalışması sona erdiğinde PHP motoru otomatik olarak açılan bağlantıları keser, hafızada kullanılan alanları temizler, kısacası yaratılan birçok şeyi otomatik olarak kendiliğinden yok eder. Ancak diyelim ki scriptin çalışma süresince açılan geçici bir dosya söz konusu ise bu dosyanın silinme işlemini bir yıkıcı fonksiyona koyabiliriz. Burada dikkat edilmesi gereken nokta yıkıcı fonksiyon içersinde herhangi bir kural dışı durum oluşturmamamızın gerektiğidir. Bu durumun oluşması halinde PHP motoru "fatal error" hatası verecektir.

# Public / Private / Protected Kelimeleri

PHP 5 ile eklenen bu kelimeler C++ dilindeki ile aynı anlamlara gelmektedirler.

## Public:

"public" kelimesi ile tanımlanan değişken ve metotlara sınıf dışından doğrudan erişilebilir. Sınıftan oluşturulmuş bir nesne doğrudan bu değişkenlere ve metotlara ulaşabilir.

## Private:

"private" kelimesi kullanarak tanımlanmış değişken ve metotlara yalnızca o sınıf içersinden ulaşılabilir. Sınıftan oluşturulmuş bir nesne doğrudan bu değişkenlere/metotlara ulaşamaz. Private kullanımı, ileri düzey hız gereksinimi olmayan projelerde güzel bir güvenlik tedbiridir. Bu değişkenin değerlerini değiştirme yetkisine sahip metotlara çeşitli denetimler eklemek mümkündür. Örnek olarak yalnızca 1-1000 arası değerler alabilecek bir değişkene asla almaması gereken bir değer verilememesi ilgili metotta kontrol ettirilebilir. Böylelikle bu değişkeni kullanan programın diğer kısımları hatalı değerlerden olumsuz olarak etkilenmezler.

## Protected:

"protected" olarak tanımlanmış değişkenler tek bir sınıf söz konusu olduğunda aynen "private" değişkenler gibi davranırlar. Bu değişkenlere ancak sınıf içersinden ulaşılabilir ve sınıf dışından tanımlanan bir nesne üzerinden doğrudan ulaşılamaz. Ancak "protected" kelimesinin farkı türetilmiş sınıflarda ortaya çıkar. Eğer B sınıfı A sınıfından türetilmiş ise A sınıfında "protected" olarak tanımlanmış değişkenlere B sınıfından ulaşılabilir.

Tüm ihtimalleri düşünecek olursak;

## Public değişkenlere:

Sınıf içersinden: Ulaşılabilir

Sınıf dışından: Ulaşılabilir

Türetilmiş bir sınıftan: Ulaşılabilir

## Private değişkenlere:

Sınıf içersinden: Ulaşılabilir

Sınıf dışından: Ulaşılamaz

Türetilmiş bir sınıftan: Ulaşılamaz

## Protected değişkenlere:

Sınıf içersinden: Ulaşılabilir

Sınıf dışından: Ulaşılamaz

Türetilmiş bir sınıftan: Ulaşılabilir

# Sınıflar İle İlgili Değerlendirme

PHP 5'de sınıflar, yukarıda anlatılan özellikler haricinde ":: operatörü", "static, self, parent kelimeleri", "overloading" gibi birçok özelliğe daha sahiptir. Ancak PHP projelerinde genel olarak ihtiyaç duyulan konular yukarıda vurgulanan konulardır. İleri derece sınıf kullanım örnekleri zaman içersinde anlatılacaktır.

# Veritabanı Uygulamaları

Bu bölüme kadar hep PHP'nin genel yapısı ile ilgili bilmemiz gerekenleri ve bunlarla ilgili çeşitli örnekler gördük. Şimdi yavaş yavaş PHP ile daha ciddi uygulamalar yazabilmek amacıyla veritabanlarıyla çalışma yöntemlerine göz atacağız.

PHP ile ODBC destekleyen herhangi bir veritabanını kolayca kullanmamız mümkündür. Ancak PHP'nin ilk yıllarından beri desteklediği yegane veritabanı MySQL'dir. PHP 4 versiyonunda genellikle PHP'nin çekirdeğinde hazır olarak gelen MySQL eklentisi PHP 5 mimarisinde yeniden dış eklenti haline getirilmiştir. Yani PHP 5'de mysql kullanabilmek için php.ini dosyasında yer alan eklentiler bölümünden (extensions) "php_mysql.dll"i aktive etmenizdir.

Bir eklenti nasıl aktif edilir? "php.ini" dosyası PHP 4'de genellikle C:\Windows ya da C:\WinNT klasöründe, PHP 5'de ise PHP'nin kurulu olduğu klasörde yer alır. Öncelikle "php.ini" dosyasının içersinde yer alan;

```
;;;;;;;;;;;;;;;;;;;;
; Dynamic Extensions ;
;;;;;;;;;;;;;;;;;;;;
```

yazan bölümü bulmanız gerekiyor. Bu bölümde birçok;

```
;extension=php_*.dll
```

şeklinde satırlar göreceksiniz. "*" karakterinin bulunduğu bölümde eklentinin ismi yazacaktır. Örnek olarak MySQL'in eklentisi php_mysql.dll'dir. Başlarında gördüğünüz ";" ini dosyalarında yorum anlamına gelir. Yani ;'den sonra ne yazarsanız yazın hiçbir işlevi olmayacaktır, sadece açıklama satırı olarak algılanacaktır. Aktive etmek istediğiniz eklentilerin başlarında yer alan ";"

karakterini silerek eklentiyi çalıştırabilirsiniz. Eğer bir şekilde hata mesajları almaya başlarsanız yine php.ini'de yer alan "extension_dir" satırını bulun. Bu satır dll'lerin bulunduğu klasörü işaret etmelidir. Örneğin eklentiler PHP 5'in kurulu olduğu klasörün içersindeki "ext" klasöründe yer alıyorsa;

```
extension_dir = "./ext/"
```

"extension_dir"ın bu şekilde tanımlanması gerekir.

"php.ini"de yapılan değişikliklerin algılanması için önemli bir husus PHP'nin kurulum biçimidir. Eğer PHP'yi CGI olarak kurmuşsanız php.ini'de yapılacak herhangi bir değişiklik bir sonraki PHP dosyası çağırıldığında otomatik olarak algılanmış olacaktır. Ancak eğer ISAPI kurulum yapmışsanız web server'ınızı durdurup başlatana kadar bu değişiklikler algılanmayacaktır.

PHP ve MySQL'in kurulumundan dolayı yaşamanız muhtemel birçok hata mesajı mevcuttur. Bu hata mesajlarını almanız halinde PHP ve MySQL forumlarında ya da daha hızlı bir şekilde Google gibi bir arama motorunda hata mesajlarını arayarak çözümlere ulaşabilirsiniz.

Şimdi MySQL veritabanına bağlanmak ve bir MySQL tablosunu listelemek ile ilgili çeşitli yöntemleri görelim.

# 1. Yöntem - Düz, Açık Kod Tarzı

```
<?php
// Veritabani Baglantisi
$baglanti = mysql_connect("127.0.0.1",
                          "root",
                          "")
    or die("MySQL'e erisilemiyor");
mysql_select_db("mysql", $baglanti)
    or die("Veritabanina erisilemiyor");

// SQL Cumlesi
$sql = "SELECT * FROM db";

// SQL'i Calistir
$tablo = mysql_query($sql, $baglanti);

// Tabloyu Ac
echo "<table border='1'>";

// Tablo Basligi
echo "<tr bgcolor='#EEEEEE'>";
for($i = 0; $i < mysql_num_fields($tablo); $i++)
    echo "<td>".mysql_field_name($tablo, $i).
         " </td>";
echo "</tr>";

// Tablo Icerigi
while($satir = mysql_fetch_array($tablo))
{
    // Tablo Sat#rlari
    echo "<tr>";
    for($i = 0;
        $i < mysql_num_fields($tablo);
        $i++)
        echo "<td>".$satir[$i]." </td>";
    echo "</tr>";
}

// Tabloyu Kapat
echo "</table>";
?>
```

Bu yöntem ile ekrana MySQL Sunucusu'nda yer alan "mysql" veritabanındaki "db" tablosunun içeriğini bir HTML tablosu şeklinde ekrana dökmüş oluruz. Aslında bu örneğin başında ve sonunda <html>, <head>, <body> gibi taglerin de yer alması gerekir ancak şu aşamada bunları kullanmadan da sonucu görebiliriz.

Burada dikkat etmeniz gereken komutlar "mysql_" ile başlayan komutlardır, yani;

- mysql_connect()
- mysql_select_db()
- mysql_query()
- mysql_num_fields()
- mysql_field_name()
- mysql_fetch_array()

komutları tabloyu listelerken kullandığımız komutlardır. Bu komutların haricinde de birçok mysql komutu mevcuttur ancak ilk aşamada bu komutlar işimizi görecektir.

# mysql_connect()

Bu komut MySQL serverına bağlantı yapmayı sağlar. Bu bağlantı olmaksızın MySQL sunucusuyla ile ilgili herhangi bir işlem yapmamıza olanak yoktur.

Bu komuttan dönen değer bağlantıya ait bir ID numarasıdır. Bu numara özellikle birden fazla MySQL bağlantısının aynı anda açılması durumunda önem kazanır. Eğer tek bir bağlantı söz konusu ise o kadar önemli değildir.

mysql_connect 5 parametre alır ancak tüm parametreleri de opsiyoneldir. Bu parametreleri kısaca tanıyalım.

- MySQL sunucu ismi/IPsi: MySQL sunucusunun Host ya da IP'si
- Kullanıcı Adı: MySQL sunucusu üzerindeki yetkili bir kullanıcı ismi
- Şifre: Kullanıcının şifresi
- Yeni bağlantı: Eğer daha önce mysql_connect komutu çağırılmışsa yeni bağlantı yapmadan önceki bağlantının ID'sini almaya yarar.
- İstemci bayrakları: Bu parametre ile MYSQL_CLIENT_SSL, MYSQL_CLIENT_COMPRESS, MYSQL_CLIENT_IGNORE_SPACE veya MYSQL_CLIENT_INTERACTIVE gibi istemci tarafında önem arz eden özellikler açılıp kapanabilir.

Yukarıdaki örnekte biz yalnızca ilk 3 parametreyi kullandık. Siz kendi test ortamınıza göre MySQL sunucusunun Host ya da IP'sini, yetkili kullanıcı ismini ve şifreyi değiştirmelisiniz. Eğer MySQL, PHP ile aynı bilgisayara kurulmuş ise mysql_connect'in ilk parametresi olarak local host ya da ip'yi girebilirsiniz ("localhost" ya da "127.0.0.1"). MySQL başka bir bilgisayarda kurulu ise o bilgisayarın networkdeki ismi ya da IPsini girmeniz gerekmektedir.

Kullanıcı adı/şifre ile ilgili karşılaşabileceğiniz en temel problem, MySQL sunucusunun PHP ile aynı bilgisayarda kurulu olmaması durumunda geçerli bir kullanıcı adı/şifre bilgisiyle bağlantı kurulamamasıdır. Büyük olasılıkla problem söz konusu kullanıcının yalnızca lokal yetkisinin oluşu, uzaktan bağlantılara izin vermeyişidir. Bu durumda MySQL üzerinde yetkili bir kullanıcı yaratmanız ve bu kullanıcıya uzaktan erişim için izin vermeniz gerekir. Bu işlem için PHPMyAdmin ya da MySQL'in kendi araçlarından birini kullanmanızı öneririm.

Bir diğer önemli nokta ise PHP ve MySQL versiyonlarınızın uyumluluğudur. Eğer PHP 5 kullanıyor ve çok eski bir MySQL sürümü kullanıyorsanız PHP'nin mysql eklentisi eski veritabanlarına erişemeyebilir. Geçmişe yönelik uyumluluk açısından PHP 5 ile birlikte "mysqli" isimli bir eklenti gelmiştir. Kullanım olarak hemen hemen aynıdır. php.ini dosyasından "php_mysql" eklentisini kapatıp "php_mysqli" eklentisini açmanız ve "mysql_" şeklinde kullandığınız tüm komutları "mysqli_" şeklinde kullanmanız gerekmektedir. Fakat benim tavsiyem, çok geçerli bir nedeniniz olmadığı sürece güncel bir MySQL versiyonu kullanmanızdan yanadır.

# mysql_select_db()

Bu komut mysql_connect komutundan sonra kullanılmalıdır. Eğer MySQL ile başarılı bir bağlantı kurulmuşsa bu durumda mysql_select_db ile MySQL üzerinde yer alan herhangi bir veritabanına erişim sağlanabilir. Bu örnekte MySQL'in içinde hazır gelen "mysql" veritabanını kullandık. Siz kendi veritabanlarınızdan biriyle de test edebilirsiniz.

mysql_select_db'den dönen sonuç boolean bir değerdir, yani true/flase şeklindedir. Ancak bu örneklerde "or die" kullandığımız için dönen değerin false olma ihtimalini kontrol etmeye gerek yok. Herhangi bir problem çıkması halinde "die" komutunda yazdığımız mesaj ekrana yazılacak ve PHP scriptinin çalışması o noktada sona erecektir.

mysql_select_db 2 parametre alır. İlk parametre mecburi, ikincisi opsiyoneldir.

- Veritabanı ismi: MySQL üzerinde bağlanılmak istenen veritabanının ismi.
- Bağlantı numarası: mysql_connect komutundan dönen numara.

Tek bir bağlantı olduğu taktirde ikinci parametre kullanılmadan da istenilen veritabanına bağlanılabilir.

# mysql_query()

MySQL'e bir sorgu gönderir. Bu komutu kullanabilmek için yalnızca MySQL'e bağlı olmak yeterlidir. Henüz bir veritabanı seçmemiş bile olsanız gönderebileceğiniz sorgular mevcuttur (örnek sorgu: SHOW DATABASES).

mysql_query bir tablo döndürür. Bu tablonun kaça kaç boyutunda olacağı sorguya göre değişir. Tablo tek bir hücreden oluşabileceği gibi binlerce/milyonlarca kolon ve satırdan da oluşabilir.

mysql_query 2 parametre alır. İlk parametre mecburi, ikincisi opsiyoneldir.

- Sorgu: MySQL sunucuna gönderilecek SQL cümlesi.
- Bağlantı numarası: mysql_connect komutundan dönen numara.

Bu komuttan dönen tablonun hücrelerine erişmek için tablo üzerinde satır satır dolaşmak ve her satırdan dönecek kolon dizilerinden hücre değerlerini okumak gerekir.

# mysql_num_fields()

Bu komut tabloda yer alan toplam kolon sayısını döndürür.

mysql_num_fields tek (mecburi) parametre alır.

- Tablo: mysql_query'den dönen tablo.

# mysql_field_name()

Tablonun kolonlarının başlıkları bu komut ile okunur. Örnek olarak veritabanında isim, soyad, şirket, telefon kolonlarından oluşan bir tablo var ise bu komut ile bu başlıkları okumamız mümkündür.

mysql_field_name komutundan text biçiminde kolon ismi döner.

mysql_field_name 2 mecburi parametre alır.

- Tablo: mysql_query'den dönen tablo.
- Kolon numarası: 0'dan başlayarak kaçıncı kolonun ismini döndüreceği.

Kolon numarası örnek olarak 4 kolonlu bir tablo için 0-3 aralığındadır.

# mysql_fetch_array()

Bu komutun görevi, tablonun satırları üzerinde ilerleyerek satır içeriklerini bir dizi halinde döndürmektir. Bu komutun bir diğer özelliği bir satırı okuduktan sonra otomatik olarak bir sonraki satıra atlamasıdır. Bu komut peş peşe çağrılarak tüm satırlar oku-

nabilir.

Dönen değer bir dizidir. Eğer tablonun sonuna gelinmiş ve okunacak satır kalmamışsa "false" değer döndürür. Bu özellik sayesinde bir döngü ile tüm tabloyu ekrana yazdırmak oldukça basit hale gelmiştir.

mysql_fetch_array 2 parametre alır. İlk parametre mecburi, ikincisi opsiyoneldir.

- Tablo: mysql_query'den dönen tablo.
- Sonuç türü: Diziyi kolon isimleri, numerik indeks ya da her ikisiyle birlikte kullanabilmek mümkündür. Bu parametre MYSQL_ASSOC, MYSQL_NUM ya da MYSQL_BOTH sabitlerine eşitlenebilir. Eğer parametreyi kullanmazsak varsayılan değeri MYSQL_BOTH'dur.

Peki hem kolon ismi hem de numerik indeks kullanabilmenin avantajı nedir? Aşağıdaki örneklerde bu avantajı gözlemleyeceğiz.

Örnek 1:

```
$sql = "SELECT isim, soyad, sirket, telefon
        FROM rehber";
$tablo = mysql_query($sql);
$satir = mysql_fetch_array($tablo);

echo $satir[3];
echo $satir["telefon"];
```

Örnek 2:

```
$sql = "SELECT telefon, isim, soyad, sirket
        FROM rehber";
$tablo = mysql_query($sql);
$satir = mysql_fetch_array($tablo);

echo $satir[3];
echo $satir["telefon"];
```

Tablonun ilk satırındaki değerlerin şu şekilde olduğunu varsayalım:

isim: Ali

soyad: Albayrak

sirket: Plazma Ltd. Şti.

telefon: +90 212 333 44 55

İlk örnekte tablonun ilk satırında 3 numaralı indekste yer alan değer ve "telefon" kolonuna ait değeri peş peşe yazdırmış oluyoruz. Satır indeksleri 0'dan başladığına göre 3 numaralı indeks yine telefon kolonuna denk gelir. Yani Örnek 1'in çıktısı şu şekilde olacaktır;

+90 212 333 44 55+90 212 333 44 55

İkinci örnekte ise 3 numaralı indeksteki telefon 0 numaralı indekse kaymış ve 3 numaralı indekste bu defa şirket yer almakta. Ekrana çıktı veren kod ise aynı yani 3 numaralı indeks ve "telefon" kolonunu yazdırıyor. Bu durumda çıktı;

Plazma Ltd. Şti.+90 212 333 44 55

olarak değişecektir.

Sonuç:

İndeks kullanımı özellikle kolonların ne oldukları çok önem teşkil etmediği ve bir loop ile tüm kolonların ekrana yazdırılacağı durumlarda doğru bir seçimdir. Ancak sabit bir tablo yapısından sabit kolon isimleriyle işlem yapacaksanız indeks yerine kolon isimlerini kullanarak yazacağınız programlarda kolon isimlerinin yer değiştirmesi gibi durumlar problem teşkil etmeyecektir.

# mysql_num_rows()

Yukarıdaki örnekte kullanmamış olsak da mysql_num_rows da çok önemli bir mysql komutudur. Bu komut bize tablodaki toplam satır sayısını verir.

Dönen değer bir sayıdır. Bir problem çıkması halinde "false" değer döndürür.

mysql_num_rows tek (mecburi) parametre alır.

- Tablo: mysql_query'den dönen tablo.

# 2. Yöntem - Sınıf Kullanımı

```
<?php
// MySQL Baglanti Sinifi
class MySQLBaglantisi
{
    // Sinif Degiskenleri
    private $baglanti;
    private $sunucu;
    private $kullaniciAdi;
    private $sifre;

    // Yapici Fonksiyon
    function __construct($_sunucu, $_kullaniciAdi, $_sifre)
    {
        // Sinifi Olustururken Ilk Degerleri Oku
        $this->sunucu = $_sunucu;
        $this->kullaniciAdi = $_kullaniciAdi;
        $this->sifre = $_sifre;
    }

    // Baglan metodu
    public function Baglan($veritabani)
    {
        // MySQL'e Baglan
        $this->baglanti = mysql_connect($this->sunucu,
                                        $this->kullaniciAdi,
                                        $this->sifre)
            or die("MySQL'e erisilemiyor");
        mysql_select_db($veritabani, $this->baglanti)
            or die("Veritabanina erisilemiyor");
        return $this->baglanti;
    }
}

// Tablo Sinifi
class Tablo
{
    // Sinif Degiskenleri
```

```
    private $html;

    // Yapici Fonksiyon
    function __construct($sql, $baglanti)
    {
        // "html" Degiskenini Temizle
        $this->html = "";

        // SQL'i Calistir
        $tablo = mysql_query($sql, $baglanti);

        // Tabloyu Ac
        $this->html .= "<table border='1'>";

        // Tablo Basligi
        $this->html .= "<tr bgcolor='#EEEEEE'>";
        for($i = 0;
            $i < mysql_num_fields($tablo);
            $i++)
            $this->html .= "<td>".
                   mysql_field_name($tablo, $i).
                   " </td>";
        $this->html .= "</tr>";

        // Tablo Icerigi
        while($satir = mysql_fetch_array($tablo))
        {
            // Tablo Satirlari
            $this->html .= "<tr>";
            for($i = 0;
                $i < mysql_num_fields($tablo);
                $i++)
            $this->html .= "<td>".$satir[$i].
                           " </td>";
            $this->html .= "</tr>";
        }

        // Tabloyu Kapat
        $this->html .= "</table>";
    }

    // Goster metodu
    public function Goster()
    {
        // Tablosu Ekrana Yansit
        echo $this->html;
    }
}

// Veritabani Baglantisi
$MySQL = new MySQLBaglantisi("127.0.0.1",
                             "root",
                             "");
$baglanti = $MySQL->Baglan("mysql");

// SQLlere Ait Tablolari Olustur
$tablo1 = new Tablo("SELECT *
                     FROM db", $baglanti);
$tablo2 = new Tablo("SELECT *
                     FROM user", $baglanti);

// Tablolari Ekranda Goster
$tablo1->Goster();
echo "<br>";
$tablo2->Goster();
?>
```

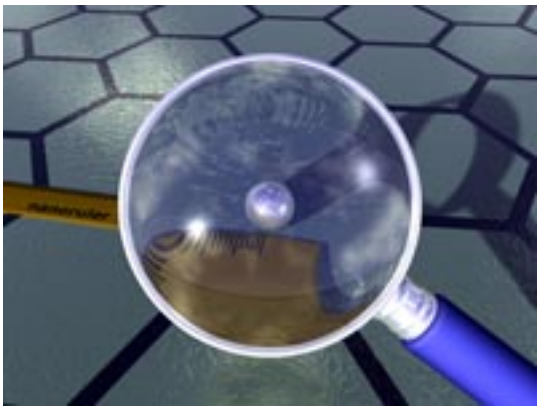
Bu örnekte daha önce öğrendiğimiz sınıflar ile ilgili öğrendiğimiz bilgileri veritabanı konusunda öğrendiklerimizle birleştirmiş olduk.

İki farklı sınıf kullandık. MySQLBaglantisi ve Tablo sınıfları.

MySQLBaglantisi sınıfı bütün MySQL bağlantı işlerini üstlenen sınıftır. Bu sınıfı yaratırken sınıfın yapıcı fonksiyonu bağlantı için gerekli parametreleri alır ve kaydeder. Baglan() metodu ise bu bilgileri kullanarak bağlantı sağlar.

Tablo sınıfı ise yine yapıcı fonksiyonundan SQL sorgusu ve veritabanı bağlantısı bilgilerini kaydeder ve tabloyu HTML kodlar halinde oluşturur. Goster() metodu ile de oluşan bu HTML kodların ekrana çıktısını vermek mümkündür.

# Bu Bölümün Değerlendirmesi



**Şekil 2.**

Geçen bölümde PHP'nin temel özellikleriyle bağladığımız bu yazı dizisinde, bu bölümde sınıflar gibi önemli ve PHP dilinde gelişmekte olan bir konuyu gördük. Daha da önemlisi PHP'nin en önemli kullanım alanlarından biri olan veritabanı uygulamalarına da giriş yaptık. Elbette ki bu doküman veritabanı konusuna derin bir şekilde değinmeyecektir ya da SQL dili hakkında detaylı bilgi vermeyecektir. Ancak çok az bir veritabanı altyapısına sahip bir kişinin PHP ile kolayca veritabanı uygulaması yazabilmesi için gerekli temel bilgileri içermektedir.

Önümüzdeki sayıda daha faydalı kullanım alanları olan örneklerle görüşmek üzere.

emir (at) akaydin (nokta) com

# "Usta... ActionScript 3.0 üzerine pişmişinden OOP kes, az da MVC döküver..."

Cem (Spaztica) Gencer

Beşiktaş'ta sabit pazarın yanında, üst geçidin altında yer alan ve yakın zamanda kapanan Bolu Et Lokantasını düşünürken attığım bu başlık, aslında tüm yazının içeriğini anlatıyor -ya da benim acıkmaya başladığımı... FutureWave Software (1993) olarak başlayan, kısa sürede (1996) MacroMedia tarafından satın alınan sonra da -hala yabancılığını çeksek de- Adobe'un gücünden beslenmeye başlayan bu platform, ağır gelişim evresini aşarak yapısının giderek kurallara bağlı hale gelmesi ve OOP gibi çağdaş bir şekle girmesiyle sağlıklı hale geldi.

Uzun süredir Flash'a göz ucuyla bakıp çeşitli denemelerinizde performansı düşük olduğu için burun kıvırıyorduk. AS2 ve öncesinde kullanıcılar için düşünülmüş esneklikler, hantallaşmaya sebep oluyordu. Kurallara ve OOP ilkelerine daha bağlı hale gelen ActionScript 3 ile artık daha erişkin bir dil / platformdan söz etmek mümkün. Yeni duyulmaya başlayan Flash 10 (Astro) dedikodularından da Flash'ın konumlanmasının radikal olarak değişeceği; C++ erişimine, 3D rotasyona ve IK (bones) ile daha pratik animasyonlar oluşturulabileceğini geçenlerde bir blogda okudum. Yani Adobe, Flash ekibinin oluşturduğu ivmeyi, sağladığı kaynaklarla da daha da artırdı. Önümüzdeki versiyonlarda bizi neyin beklediğini kestirebilmek giderek zorlaşacak. Flex ve Air gibi yan teknolojilerle, Spry ve yakında alınacağını duyduğum AMFPHP gibi destekleyici framework'lerle Flash'ın gücü gerçekten katlanacağa benziyor. Karşı cephede MacroFos'un oluşturmaya çalıştığı SilverLight gibi bir ürünün zamanın getirdiği bu deneyimle baş etmesi ve sağlıklı bir yer edinmesi zor olacağa benzer. Kurumsal müşteriler, elbette her zaman uyuşturucu bağımlısı gibi kendi torbacılarından beslenmek isteyeceklerdir. Neticede bir torbacı da olsa, müşterisini kaybetmek istemez.

Bu yazıda OOP ilkelerinin ve MVC'nin AS3 üzerinde nasıl uygulanabildiğini göstermeye çalışacağım. AS2'den AS3'e geçmek isteyenler de buradaki temel prensiplere sadık kalırlarsa sorunları azalacaktır. Bu temeller olmadan önümüzdeki sayılarda ele alacağım konuları kavramak zor olurdu. Görsel efekt ve benzeri demo kültürünü ilgilendiren konular ağırlıklı ilgi alanımız olacak; basit bannerlar yapmak ya da dinamik içerik sunmak gibi konularda web üzerinde yazılmış tonla kaynak zaten mevcut.

Yazılarım, terminoloji açısından klasik bilişim çevirilerinden biraz farklı olabilir. Object Oriented Programming tanımlamasını herkesin kabul ettiği gibi nesne yönelimli diye çeviremiyorum; bunun önemli sebebi pratiklik. İngilizce terminoloji halinden fazla uzaklaşmazsak, başka bir yerde object olarak okuduğunuz bir tanımlamanın nesne olup olmadığını düşünmezsiniz. Bunun gibi bazı terimleri ingilizce bırakmak okunurluğu rahatlatan, uygulama açısından da pratik olacak bir tercih.

## Kontrol Etmek, Kontrol Edilmektir...

Bir Zen ustası kontrolü reddeder. Kontrol etmeye çalıştığınız sürece kontrol sizi kendisine bağlı kılar. Kodunuzu tek bir yerde toplamak, kontrolü elinizde tutmak gibidir. O güç asla sizde olmayacaktır, asla kontrolü elinizde tutamazsınız, tuttuğunuzu düşündüğünüzde aslında siz kontrol altında tutuluyor olursunuz. Kontrol de asla ulaşılamayan gereksiz bir amaç olur. Budistlerin Samsara olarak adlandırdığı ve farkındalıkla kurtulmaya çalıştıkları sonsuz döngü ancak farkındalık ve dışında durmayı tercih etmekle kırılabilir; aynı şekilde kontrol döngüsü de.

Düz akışlı ve fonksiyon bazlı kodlamada alıştığımız herşeyi bir dosyada, bir listede toplamak, aşırı kontrolün sonucu; yazdığımız programa ait her parçanın elimizin altında olmasını isteriz. OOP'un oluşmasıyla, her oluşturulan objeyi bir yaşam biçim olarak görebiliriz. Bu objelere verdiğiniz görevlere, yapacaklarına güvenmeniz, işleri bu şekilde bölüştürüp delege etmeniz, vaktinizi artıracak, karışıklığı ise azaltacak bir yol.

Yazdığınız kod parçalarını amaçlarına göre düzenleyin ve her bir metodu, gerçek amaçları için kullanarak kendinizi özgür bırakın. Amaçları iç içe geçmiş kod yığınları sadece kodunuzu karıştırmaya ve sizi bug'larla uğraşmaya mahkum bırakır.

## Hantallık ya da Aşırı Esneklik...

ActionScript'in önemli bir yavaşlatıcı unsuru, esnek model idi. Bunun sayesinde programlama deneyimi fazla olmayan kişiler, rahatlıkla 3-4 satır ekleyerek pratik fonksiyonları gerçekleştirebiliyordu. En basitinden değişkenlerin tipinin tanımlanmamış olması, player'ın her türlü değişkeni kendi içinde inceleyerek uygun şekilde ele alması gibi bir yavaşlatıcı etkene sebep oluyordu. Bu yüzden AS3 kullanırken değişken tiplerini tanımlamak bir gereklilik oldu. Daha önceden kolaylık sağlayan Number tipi de int ve uint gibi iki gruba ayrıldı. Bu sayede int ile poiztif ve negatif değerleri, uint ile de sadece pozitif değerleri tanımlıyoruz.

uint'in avantajı, negatif değerleri göstermek için bir bit harcamadığından daha büyük numaraları (int'in 2 katı) tanımlamakta kullanılabiliyor oluşu. Number'da olduğu gibi player bu değişkenin içerik tipini algılamaya çalışmadığından daha hızlı işlem yapabilecektir.

# Erişim

Şimdiye kadar az-çok Flash ile uğraşan herkes aşağıdakine benzer bir kodu MovieClip'e kod ekleyerek bununla timeline'ın akışına müdahale etmişlerdir.

```
_parent.gotoAndStop(2);
```

Bunu AS3 ile çalıştırmaya kalktığınızda compiler tonla hata kusacaktır. Sebebi, AS3'ün artık class inheritance şeklinde çalışmasıdır. AS2 prototype inheritance temelli olduğu için bu gibi pratik müdahaleler kolaylıkla yapılabiliyordu. AS3'e geçerken bunu kurallara uygun hale getirmek için

```
var myParent = parent;
myParent.gotoAndStop(2);
```

şeklinde yazmamız, problemi çözecektir. Burada myParent ile yeni bir obje tanımlayarak erişimimizi onun üzerinden yapıyoruz. Elde etmek istediğimizi bu şekilde çok az değiştirerek OOP kurallarına tam bir uygunluk sağladığımızda AS3'e geçişte problemlerimiz de azalacaktır. Aynı yöntemle String ve fonksiyonlarda da değişken tipini önden tanımlamamız, daha hızlı ve hatasız çalışmamızı sağlayacaktır.

```
var numOut:Number = 15;
var numOut:int = 15;
------------------------------
var slowVar = "myString";
var fastVar:String = "myString";
------------------------------
function slowFunc()
{
    return 100;
}
function fastFunc():Number
{
    return 100;
}
```

# Class

AS3'de görsel arabirim dahil herşey class üzerinden tanımlanmıştır. Flash'a ait temel class'lar, compile aşamasında otomatik olarak movie içine dahil edilir.

Class'ları oluştururken package ile oluşturacağımız metodları gruplayabiliriz:

```
package
{
    public class Circle
    {
    ...
    }
}
```

Temel class'lar dışında bir class kullandığınızda onu compiler'a tanıtmazsanız kullanamazsınız. Kullandığınız class'ları kullandığınız yerde import etmeniz -temel class'lardan olsalar da- temel bir alışkanlık olmalı. Bunun için import komutunu kullanıyoruz:

```
package
{
    import flash.display.MovieClip;
    public class Circle
    {
    ...
    }
}
```

Timeline üzerinde class import ederken birden fazla yerde aynı class'ı kullanacaksanız ya her kullanıldığı yerde import etmeniz ya da aşağıdaki gibi class'ı oluştururken erişim yolunu da tanımlamanız gerekiyor:

var myFoo:foo = new macr.util.foo();

# Public / Private

Günün birinde bir programcı, dünyadaki tüm edebiyatı, görsel sanatları, sayıları ve istatistikleri bir araya getiren sonsuz bir veritabanı oluşturdu. Sonra buna her türlü sorgulamayı kolaylıkla yapabileceği bir sorgulama sistemi tasarladı ve yazdı. Ondan sonra da çalışmasının meyvelerini tadabilmek için bilgisayarının karşısına oturdu.

3 dakika sonra başı ağrımaya başladı. 3 saat sonra rahatsızlık duymaya ve kötüleşmeye başladı. 3 gün sonra, tüm veritabanını imha etti. Sebebini sorduklarında "Bu sistem tüm verileri önüme getiriyor, herşeyi incelememi ve sorgulamamı sağlıyordu. Her türlü bilgiye erişebiliyor, her şeyi görebiliyordum, her yeri gezebiliyordum. Bu, bilinebilecek her şeyi öğrenmemem için bahanelerimi ortadan kaldırıyor. Yemek yiyemez, uyku uyuyamaz hale geldim. Tek yapabildiğim, bu sonsuz veritabanında gezinmekti. Artık dinlenebilirim."

Sonsuz açıklık ve erişilebilirlik, sağlıklı değildir. Kodunuza ait her şeyin açık ve erişilebilir olması, müdahalelerin yolunu açar. Sizin planladığınızın dışında müdahaleler de kodunuzun sağlıksız çalışmasına yol açar.

Public ile bir metodu ya da property'yi dışarıdan erişime açık tutmak istediğimizi belirtebiliriz. Bunun tersi Private özellikle class iç hesaplamalarında kullanılabilecek metod / property'ler içindir.

# Property

AS3 öncesi property dediğimiz objelere ait değerlere erişim için

```
myClip_mc._alpha = 50;
```

şeklinde önünde _ işaretli bir yaklaşım uyguluyorduk. AS3 ile hem bu öndeki _ işareti kalkıyor, hem de 0-100 arası değerler artık 0...1 aralığında olacak şekilde veriliyor. AS2'ye kadar yüzde şeklinde cinsinden değer belirtmek, player'ın ilgili property değeri üzerinde çarpma ve sonra 100'e bölme işlemi yapmasını, dolayısıyla da hızın azalmasına sebep oluyordu.

# Event

AS3 öncesi bir movieclip'i mouse ile etkileşimini sağlamak çok pratik idi; ilgili movieclip'in koduna şu türden bir kod eklemekle Flash'ın Button tanımlaması da tarih olmaya başlamıştı:

```
on (release){
    doSomething();
}
```

AS2 ile bu yöntem, timeline'da yazılabilecek şu şekilde bir kodla da sağlanabiliyordu:

```
myButton.onRelease = function(){
    doSomething();
}
```

Ama artık AS3 ile herşey kurallara çok uygun olmak durumunda. Objeler arası etkileşim, bunların birbirini dinlemesi ve buna uygun tepki göstermesiyle olabilir. Bu yüzden listener (dinleyici) oluşturup istediğimiz durum gerçekleştiğinde bir işleyiş oluşturabiliriz.

```
function ClickHandlerFunction(event)
{
    doSomething();
}
myButton.addEventListener(
    MouseEvent.CLICK, ClickHandlerFunction);
```

Aslında yaptığımız, klasik bir fonksiyon tanımlamasında parametre olarak bir event objesi getirecek şekilde dönüştürerek bunu bir dinleyiciye bağlamak. MouseEvent.CLICK, dinlenecek durum tipini tanımlanmasını sağlıyor. Fonksiyonda kullanmayacak olsanız bile parametre olarak event objesini belirtmeniz gerekiyor, yoksa durum gerçekleştiğinde parametresiz tanımlanmış fonksiyona parametre geçmeye çalıştığınız için hata verecektir. event objesini incelediğinizde bunun, durumu gerçekleştiren objeye refere eden bir obje olduğunu görürsünüz. Farklı durumları tek bir fonksiyon ile algılamak istediğinizde bu event objesini kontrol ederek öğrenebilirsiniz.

# Inheritance

Gerçek bir OOP, class'ların birbirinden türetilebildiği bir kalıtım sistemini de barındırır. Kendisini oluşturan class'a ait metod ve property'lerin devralınmasına inheritance (kalıtım) denmektedir. Bu yaklaşım, var olan çeşitli class'ları kendi metodlarımızla geliştirmemizi sağlıyor. Önceden prototype ifadesi ile Array, Math, MovieClip gibi çok kullanılan class'lara ek fonksiyonlar tanımlıyorduk. AS3 artık prototype yaklaşımı yerine class inheritance kullandığından prototype genişletme yönteminin pabucu dama atıldı.

```
public class Circle extends MovieClip
{
...
}
```

şeklinde bir class tanımlaması, MovieClip class'ını genişleterek ona Circle isimli bir metod ekleyecektir. Aşağıdaki örnekte Circle metodu, MovieClip class'ına yeni bir fonksiyon olan daire çizmeyi ekliyor. radius ve fillColor, dışarıdan erişilebilen property'ler ve fonksiyon, bunları parametre olarak alarak daireyi MovieClip class'ının drawCircle() metodu ile çiziyor.

```
package
{
    import flash.display.MovieClip;
    public class Circle extends MovieClip
    {
        public var radius = 10;
        public var fillColor = 0xFF0000;
        public function drawTheCircle()
        {
            this.graphics.clear();
            this.graphics.beginFill(fillColor);
            this.graphics.drawCircle(0 , 0, radius);
        }
    }
}
```

Timeline'da kullanılacağı zaman

```
import Circle;
var myCircle = new Circle();
myCircle.radius = 20;
myCircle.fillColor = 0x0000FF;
myCircle.drawTheCircle();
```

yazmak yeterli olacaktır.

# Static

Class'ın her yeni kullanımında değişmeyecek değerleri ve metodları Static kelimesi ile tanımlayabiliriz. Bu şekilde statik yapılan öğe, class'dan oluşturulan her çocuk objede ayrı ayrı tutulmaz, tek değer/metod tüm o class'ın çocukları için geçerli olur. Bir obje yaratıldığında, class'da Static olmayan tüm property ve metotlar, her çocuk objeye kopyalanır. Ama Static tanımlanan property ve metotlar, sadece class'a ait olurlar, onun çocuklarına aktarılmazlar.

Statik ögeleri, bir class'dan kaç defa türetildiğini kontrol etmek için kullanabilirsiniz. Buna örnek olarak bir slideshow indeksini verebilirim. İleri ve geriye gidildiğinde kaçıncı resimde olduğumuz ve başa / sona ulaşıp ulaşmadığımızı, bu değerleri statik yaparak kontrol edebiliriz.

```
class as_scripts.SlideShow3 {
    public function SlideShow3 ()
    {
    }
    public static function lMovie(s_movie:String)
    {
        var m_empty:MovieClip =
            _root.createEmptyMovieClip(
            "m_empty", 100);
        m_empty._x = 100;
        m_empty._y = 100;
        m_empty.loadMovie(s_movie);
    }
}
```

Bu class üzerinden türetilen her objede IMovie metodu var gibi gözüküyor. Ama normal şekilde

```
import as_scripts.*;
var a:SlideShow3 = new SlideShow3();
a.lMovie("../images/pic_1.jpg");
```

şeklinde bir kullanımda "error message: Static members can only be accessed directly through classes." gibi bir hatayla karşılaşırsınız.

SlideShow3 class'ının IMovie metoduna erişmek için.

```
import as_scripts.*;
SlideShow4.lMovie("../images/pic_1.jpg");
```

şeklinde doğrudan o class'a erişmeniz gerekiyor. Aynı şekilde bir property'yi de Static tanımlamanız gerektiğinde

```
class as_scripts.SlideShow3right {
    public static var s_movie:String =
       "../images/pic_1.jpg";
    public function SlideShow4right ()
    {
    }
    public static function lMovie () {
        var s_mymovie:String = s_movie;
        var m_empty:MovieClip =
            _root.createEmptyMovieClip(
            "m_empty", 100);
        m_empty._x = 100;
        m_empty._y = 100;
        m_empty.loadMovie(s_mymovie);
    }
}
```

ile değişkeni static olarak tanımlamalısınız. Aksi halde statik olan metodumuz, statik olmayan (ve her kopyada yer alan) değişkene erişmeye çalıştığında "error message: Instance variables cannot be accessed in static functions." şeklinde bir hata verecektir.

# Get / Set

OOP'un güzel yanı, oluşturduğumuz objelerin kapalı birer kutu gibi davranabilmesi ve yazan kişinin izin verdiği ölçüde dışarıya bilgi vermesidir. Class'ın içinde çok karışık hesaplamalar yapabiliriz. İçerde onlarca, hatta yüzlerce değişken kullanabiliriz, ama bu her değişkeni, her an erişime sunabileceğimiz demek değildir. Daha önceden kullandığımız getProperty ve setProperty metodları class'ımızda çalışacaktır. Ama OOP'un temelinde bir değişkene erişmek istiyorsak, ona erişen bir getter ya da onu değiştiren bir setter fonksiyon yazmak daha düzenli ve kontrollü olmamızı sağlar. Getter, değer döndüreceği için return kullanmamız kaçınılmaz. Aynı zamanda metoda :Number tanımlamasıyla bir rakam döndüreceğini belirtebiliriz. :Void ile setter metodunun herhangi bir geri dönüşü olmayacağını belirtiriz.

```
class as_scripts.Adjust_xy {
    private var nu_xDistance:Number;
    private var nu_yDistance:Number;
    private var mo_fclip:MovieClip;
    private var m_clip:MovieClip;
    public function Adjust_xy (mo_fclip)
    {
        m_clip = mo_fclip;
    }
    function get Fix():Number
    {
        return m_clip._x;
    }
    function set Fix(nu_xDistance):Void
    {
        m_clip._x = nu_xDistance;
    }
    function get Fiy():Number
    {
        return m_clip._y;
    }
    function set Fiy(nu_yDistance):Void {
        m_clip._y = nu_yDistance;
    }
}
```

# MVC

Zen, daha üst seviyede algılamakla ilgilidir. Bir bayrak dalgalandığında herkes onun dalgalandığını görür. Bir Zen öğrencisi ise bayrağı dalgalandıran rüzgarı algılar. Daha ileri seviyedeki bir Zen öğrencisi ise ne bayrak, ne de rüzgarı dikkate alır, onu algılayışımızın düşünme biçimimizle ilgili olduğunu görür. Hayat içerisindeki acı ve sevinçlerimizin düşünme biçimimizin yarattığı prangalarımız olduğu Budist düşünüş biçiminin temellerini oluşturur.

MVC'yi programlamanın Zen'i olarak görebiliriz. Karışık yapıları yüzünden bize acı çektiren bir çok hata prangasından kurtulmak da daha üst bir bakış açısıyla olabilir. Zen'i kavramanın bir kitap okumakla olamayacağı gibi MVC'nin de kemikleşmesi bol pratik yapmakla ve bu yaklaşımı sürekli hale getirmekle ilgili. Aslında Zen gibi, MVC de çok basit ve herkesin kolaylıkla kullanabileceği bir yapıdır. http://en.wikipedia.org/wiki/Design_pattern_(computer_science) başlığında benzer çeşitli yazılım şablonlarını bulabilirsiniz.

MVC'nin açılımı Model-View-Controller'dır. Gösterim, veri ve denetleme mekanizmalarını birbirinden ayrı tutarak daha odaklı halde kod yazmamızı sağlayan bu yaklaşım, daha organize çalışmamızı sağlar. Model, veritabanı, veri alışverişi gibi bilginin kaydedildiği ve okunduğu program kısımlarını içerir. View, içeriğin görselleştirildiği, kullanıcıyla iletişimin kurulduğu, front-end olarak düşünebildiğimiz bölümlerdir. Controller ise bu ikisi arasında durarak hangi durumda hangi veriyi getirileceğini, hangi gösterim şablonlarının kullanılacağını belirleyen mekanizmadır.

Verinin geçireceği dönüşümler de genellikle Controller içerisinde tanımlanır. MVC, özellikle ölçeklenebilir yapılarda işimizi rahatlatan yapısal bir tasarımdır. Daha modüler bir yapı, özellikle iş bölümü yapılmasında ve herkesin bir bölüme odaklanmasında rahatlık sağlar. Kodu ayrıştırdığı için kısmi müdahaleler ya da yapının başka bir zaman değiştirilmesi, radikal bir çalışma olmaz. Yapıyı yeni bir veritabanına adapte etmek, farklı bir görsel arabirim hazırlamak ya da işlevsel kısımları geliştirmek çok daha pratik gerçekleşir.

Aşağıdaki basit örnek, dört dosyadan oluşuyor. Dosya düzeni de aşağıdaki gibi olmalı. mvctest.fla'yı compile edip çalıştırdığınızda controller, yeni bir model yaratacak ve ardından mvctest.fla üzerinden view'ın out() fonksiyonu çalıştırılacak. Görüldüğü gibi çok basit halde verinin Model ile, ekrana çıkışın View ile, genel kontrolün de Controller ile yapılmasını sağlıyoruz. Bunu geliştirip istediğimiz uygulamaları veri alışveriş ve ekran şablonu sistemine oturtup Event sistemini de devreye alarak çeşitli durumlara göre kendi kontrollerini idare eden bir yapıya dönüştürebiliriz.

bu örneğin dosyalarını http://www.obsesif.net/files/mvc.rar adresinden çekebilirsiniz

```
mvc/
    com/
        mvctest/
            Controller.as
            Model.as
            View.as
mvctest.fla
```

```
/* ===================================== */
/* dosya ismi: mvctest.fla               */
/* ===================================== */
import com.mvctest.*;
var mc = new Controller();
mc.out();

/* ===================================== */
/* dosya ismi: com/mvctest/Controller.as */
/* ===================================== */
package com.mvctest {
    import com.mvctest.*;
    public class Controller {
        private var dataset;
        public function Controller ( ) {
            this.dataset = new Model();
        }
        public function out ( ) {
            var viewset = new View(
                this.dataset.getCoords( ) );
        }
    }
}
/* ===================================== */
/* dosya ismi: com/mvctest/Model.as      */
/* ===================================== */

package com.mvctest {
    public class Model {
        private var xPos:Number;
        private var yPos:Number;
        public function Model ( ) {
            this.xPos = 100;
            this.yPos = 100;
        }
        public function getCoords ( ):String {
            return "xPos: " + this.yPos
                + ", yPos: " + this.yPos;
        }
    }
}
/* ===================================== */
/* dosya ismi: com/mvctest/View.as       */
/* ===================================== */
package com.mvctest {
    public class View {
        public function View( content:String ){
            trace( content );
        }
    }
}
```

# Amiga Assembly Kursu - 2

Şemseddin 'Endo' Moldibi

Amiga Assembly Kursumuzun 2. bölümüne hoş geldiniz. Önceki sayımızda sizin de fark etmiş olacağınız gibi, bu yazı programcı olmayanlara değil, daha çok, programcılığı (Assembly, C veya en az bir programlama dilini) biliyor olup Amiga'da 68000 Assembly ile program yazmak isteyenlere yönelik. Ancak bu, programlamaya yeni başlayanları korkutmasın, bu şekilde öğrenmek zor olduğu kadar zevklidir de.

Önceki sayımızda 68000 register' larını, veri transfer komutlarını ve Amiga'nın hafıza yapısı konuları incelemiş, basit bir kodu nasıl derleyeceğimizi öğrenmiştik. Bu sayımızda 68000 komutlarını öğrenmeye devam edecek, donanım register' larını kısaca tanıyacağız, ardından çeşitli örnek kodları inceleyeceğiz, son olarak biraz daha görsel sonuçlar görebileceğimiz ilginizi çekeceğini umduğum bir örnek kod ile devam edeceğiz. Bunu yaparken henüz öğrenmediğimiz bazı konulara girmek zorunda kalacağız, ancak ileriki sayılarda bu konuların tümüne sırasıyla değineceğiz.

Öncelikle örnekleri kolaylıkla test edebilmeniz için hazırladığım asmpro.adf isimli dosyayı; eğer gerçek bir Amiga kullanıyorsanız boş bir diskete yazın (bunu yapan programları Internet'ten bulabilirsiniz), eğer WinUAE Amiga emulatörü kullanıyorsanız ayarlarınızı aşağıdaki gibi yapın, df0'a asmpro.adf dosyasını gösterin, ayarlarınızı kaydedin ve emulatörü başlatın. Sistem bu disketten boot ederek açılacak ve karşınıza AsmPro çıkacaktır.

WinUAE ayarlarınızda 68020 işlemci, AGA chipset (en azından ECS), 3.1 veya 2.0 kickstart rom dosyası ve ~4 MB fastram seçin.

Kodlarınızı rahatça yükleyip kaydetmek için bir harddisk oluşturmanızı tavsiye ederim. Hatta asmpro.adf disketindeki dosyaları harddiskinize kopyalayıp oradan da çalışabilirsiniz.

AsmPro ilk açıldığında gelen ekranda kullanacağınız ram tipi ve miktarını seçebilirsiniz. Şimdilik ~100 KB CHIP RAM seçebilirsiniz.

## 68000 BCC Komutları (Dallanma Komutları)

68000 işlemcilerde genel olarak CC (Conditional Code) komutları adı verilen bir grup komut vardır. Bunlar çoğunlukla program içinde dallanmaya yarayan BCC komutlarıdır:

## BCC Komutları (Koşullu Dallanma Komutları):

BCC komutlarının genel kullanım şekli aşağıdaki gibidir:

```
Bcc.S <label>
Bcc.W <label>
```

Burada komutu takip eden .S ve .W sonekleri (suffix) dallanmanın mesafesine göre belirlenir; .S kısa mesafe (short range: -128, +127 byte), .W ise uzun mesafe (word range: -32768, +32767 byte) dallanma anlamındadır.

Normalde belirtmediğiniz takdirde derleyiciniz en uygun olanı seçecektir.

BCC Komutları listesi aşağıdadır:

```
BCS Carry Bit Set
BLS Lower or Same
BEQ EQual (Z-bit Set)
BNE Not Equal (Z-bit Clear)
BHI HIgher than
BCC Carry Bit Clear

BPL PLus (N-bit Clear)
BMI Minus (N-bit Set)
BVC V-bit Clear (No Overflow)
BVS V-bit Set (Overflow)
BRA BRanch Always

BLT Less Than
BLE Less than or Equal
BGT Greater Than
BGE Greater than or Equal
```

İngilizce açıklamalarından komutların işlevlerini kolayca çıkarabilirsiniz. Burada dallanma komutlarına genel bir isim olarak kullanılan BCC ile dallanma komutlarından biri olan BCC arasındaki benzerlik aklınızı karıştırmasın.

Bu listedeki tüm komutlar, .S ve .W eki ile kullanılabilir, .S hafızada daha az yer kapladığından ve daha hızlı olduğundan mümkün olan her durumda tercih edilir.

NOT: 68000 işlemcilerin üst modellerinde (68020+) .L eki ile 32 bitlik mesafelerde dallanma yapılabilir. Ancak 68000 işlemci de .L ekli BCC komutları desteklenmemektedir.

```
waitclick:
        btst        #6,$bfe001
        bne.s        waitclick
        rts
```

Ayrıca DBcc (Decrement and branch conditionally) komutlarını da bu listeye eklemeliyiz. DBcc komutları (DBeq, DBne, DBpl vb.) verilen data register'ını bir azaltır ve koşul SAĞLANMADIĞI

sürece veya verilen data register'ı sıfırın altına düşmediği sürece dallanma işlemini gerçekleştirir.

```
start:
        moveq         #4,D0
loop:
        nop          ;buraya kodlar girilir
        dbeq          D0,loop
        rts
```

Burada BEQ komutu ile DBEQ komutu arasındaki önemli farka değinmemiz gerek: BEQ komutu işlemin sonucu sıfır İSE dallanır, DBEQ komutu ise işlemin sonucu sıfır olana dek dallanır, sıfır olduğunda ise döngüden çıkılır.

NOT: Önce data register'ının değerinin sıfır olup olmadığına bakılır, sonra değer bir azaltılır, ilk karşılaştırmanın sonucu sıfır değilse dallanma gerçekleşir. Yani DBEQ komutundan çıkıldığında D0 register'ında $FFFF değeri olacaktır.

NOT 2: Verilen register'ın sıfır ile karşılaştırma işlemi 16 bit (word) olarak yapılır ve register'ın değeri yine word olarak azaltılır. Sayaç register'ındaki değer başlangıçta $12340000 ise DBcc komutundan sonra $1234FFFF değeri olacaktır.

Ayrıca yukarıdakilere ek olarak DBF ve DBRA komutlarını söyleyebiliriz, bu iki komut koşula (condition) bakmaksızın sadece register'ın değeri sıfır olana dek dallanma işlemi yapar. Genel olarak koşulsuz döngüler için kullanılırlar.

# Hardware Registerlar'ı

Hardware register'ları işlemcinin doğrudan custom chiplere ulusabilmesini sağlayan adreslerdir. RAM ve ROM'da bulunmazlar. Normal şartlarda, eğer bir utility veya sistem uyumlu, multitasking bir uygulama geliştiriyorsanız bu adreslere doğrudan erişmeniz gerekmez, hatta erişmemeniz daha doğru olur. Bu adreslere dolaylı olarak ulaşmanızı sağlayacak kütüphaneler vardır. Kütüphaneler konusuna ileride detaylı olarak değineceğiz.

Ancak demo (ve hatta çoğu oyun) programları bu adreslere doğrudan erişim yapılarak yazılır. Bu bölümde ileride çok defa karşılaşacağımız bu adresler hakkında kısa bilgiler vereceğim:

Hardware register'larının 68000 üzerindeki base (taban) adresi $DFF000'dır.

Yani verilen chip adreslerini bu taban adrese ekleyerek ulaşacağınız adresi bulabilirsiniz.

68000 adresi = (chip adresi) + $DFF000

Örneğin Joystick/Mouse bilgisini okumak için kullanılan adres $DFF00A için genelde sadece $00A register'ı denir.

Aşağıdaki listede donanım register'larının adreslerini ve kısa açıklamalarını bulabilirsiniz. Listede R/W olarak verilen kısım register'ın sadece okunabilir ya da sadece yazılabilir olduğunu ifade eder.

W write-only, yalnız yazılabilir; R read-only, yalnız okunabilir; ER early-read DMA için kullanılır; S strobe yazma işlemi yapıldığında bir olay tetiklenir.

Bu register'ları kullanılırken okunabilir/yazılabilir olmalarına dikkat etmelisin, asla okunabilir bir regsiter'a veri yazmaya kalkmayın ya da yazılabilir bir register'ı okumayın. Bu makinenin kilitlenmesine sebep olur.

# Hardware Register'ları Listesi

BLTDDAT (000,ER) : Blitter RAM'e veri kopyalarken kullanır. Kullanmayınız

DMACONR (002,R) : DMA kontrol, tüm DMA durumlarını bu adresten okuyabilirsiniz

VPOSR (004,R) : Dikey tarama MSB (most significant bit)

VHPOSR (006,R) : Yatay/Dikey tarama pozisyonu

DSKDATR (008,ER) : Disk DMA tamponu. Kullanmayınız

JOY0DAT (00A,R) : Joystick-mouse 0 verisi (dikey, yatay)

JOY1DAT (00C,R) : Joystick-mouse 1 verisi (dikey, yatay)

CLXDAT (00E,R) : Sprite/görüntü çarpışma verisi (oku ve temizle)

ADKCONR (010,R) : Ses, disk kontrol register'ı

POT0DAT (012,R) : Pot sayacı (dikey,yatay)

POT1DAT (014,R) : Pot sayacı (dikey,yatay)

POTGOR (016,R) : Pot port data

SERDATR (018,R) : Seri port data ve durumu

DSKBYTR (01A,R) : Disk data byte'ı ve durumu

INTENAR (01C,R) : Interrupt enable bitleri

INTREQR (01E,R) : Interrupt request bitleri

DSKPTH (020,W) : Disk adres pointer'ı (üst 3 bit)

DSKPTL (022,W) : Disk adres pointer'ı (alt 15 bit)

DSKLEN (024,W) : Disk DMA data uzunluğu (word)

DSKDAT (026,W) : Disk DMA data yazma

REFPTR (028,W) : Kullanmayınız. (Refresh pointer)

VPOSW (02A,W) : Dikey tarama MSB (yaz)

VHPOSW (02C,W) : Yatay/Dikey tarama pozisyonu (yaz)

COPCON (02E,W) : Coprocessor (Copper) kontrol register'ı

SERDAT (030,W) : Seri port data ve stop biti (yaz)

SERPER (032,W) : Seri port periyod/kontrol

POTGO (034,W) : Pot port data yaz/başla

JOYTEST (036,W): ???

STREQU (038,S) : Yatay senkronizasyon strobe adresi

STRVBL (03A,S) : Yatay senkronizasyon strobe adresi

STRHOR (03C,S) : Yatay senkronizasyon strobe adresi

STRLONG (03E,S) : Yatay senkronizasyon strobe adresi

BLTCON0 (040,W) : Blitter kontrol register'ı 0

BLTCON1 (042,W) : Blitter kontrol register'ı 1

BLTAFWM (044,W) : Blitter mask değeri (Kaynak A, ilk word)

BLTALWM (046,W) : Blitter mask değeri (Kaynak A, son word)

BLTCPTH (048,W) : Blitter pointer C (üst 3 bit)

BLTCPTL (04A,W) : Blitter pointer C (alt 15 bit)

BLTBPTH (04C,W) : Blitter pointer B (üst 3 bit)

BLTBPTL (04E,W) : Blitter pointer B (alt 15 bit)

BLTAPTH (050,W) : Blitter pointer A (üst 3 bit)

BLTAPTL (052,W) : Blitter pointer A (alt 15 bit)

BLTDPTH (054,W) : Blitter pointer D (hedef) (üst 3 bit)

BLTDPTL (056,W) : Blitter pointer D (hedef) (alt 15 bit)

BLTSIZE (058,W) : Blitter boyut/başla (genişlik, yükseklik)

BLTCMOD (060,W) : Blitter modulo C

BLTBMOD (062,W) : Blitter modulo B

BLTAMOD (064,W) : Blitter modulo A

BLTDMOD (066,W) : Blitter modulo D

BLTCDAT (070,W) : Blitter kaynak C data register'ı

BLTBDAT (072,W) : Blitter kaynak B data register'ı

BLTADAT (074,W) : Blitter kaynak A data register'ı

DSRSYNC (07E,W) : Disk okuması için sync patern register'ı

COP1LCH (080,W) : Coprocessor lokasyonu 1 (üst 3 bit)

COP1LCL (082,W) : Coprocessor lokasyonu 1 (alt 15 bit)

COP2LCH (084,W) : Coprocessor lokasyonu 2 (üst 3 bit)

COP2LCL (086,W) : Coprocessor lokasyonu 2 (alt 15 bit)

COPJMP1 (088,S) : Coprocessor restart (ilk lokasyon)

COPJMP2 (08A,S) : Coprocessor restart (ikinci lokasyon)

COPINS (08C,W) : Coprocessor tarafından kullanılır

DIWSTRT (08E,W) : Görüntü alanı başlangıcı (sol üst dikey-yatay pozisyon)

DIWSTOP (090,W) : Görüntü alanı bitişi (sağ alt dikey-yatay pozisyon)

DDFSTRT (092,W) : Data fetch başlangıç (yatay pozisyon)

DDFSTOP (094,W) : Data fetch bitiş (yatay pozisyon)

DMACON (096,W) : DMA kontrol (yaz)

CLXCON (098,W) : Çarpışma kontrol register'ı

INTENA (09A,W) : Interrupt enable bitleri

INTREQ (09C,W) : Interrupt isteği

ADKCON (09E,W) : Audio, disk, UART kontrol

AUD0LCH (0AO,W) : Ses kanalı 0 DMA data'sı (üst 3 bit)

AUD0LCL (0A2,W) : Ses kanalı 0 DMA data'sı (alt 15 bit)

AUD0LEN (0A4,W) : Ses kanalı 0 uzunluk

AUD0PER (0A6,W) : Ses kanalı 0 periyot

AUD0VOL (0A8,W) : Ses kanalı 0 ses

AUD0DAT (0AA,W) : Ses kanalı 0 data

AUD1LCH (0B0,W) : Ses kanalı 1 DMA data'sı (üst 3 bit)

AUD1LCL (0B2,W) : Ses kanalı 1 DMA data'sı (alt 15 bit)

AUD1LEN (0B4,W) : Ses kanalı 1 uzunluk

AUD1PER (0B6,W) : Ses kanalı 1 periyot

AUD1VOL (0B8,W) : Ses kanalı 1 ses

AUD1DAT (0BA,W) : Ses kanalı 1 data

AUD2LCH (0C0,W) : Ses kanalı 2 DMA data'sı (üst 3 bit)

AUD2LCL (0C2,W) : Ses kanalı 2 DMA data'sı (alt 15 bit)

AUD2LEN (0C4,W) : Ses kanalı 2 uzunluk

AUD2PER (0C6,W) : Ses kanalı 2 periyot

AUD2VOL (0C8,W) : Ses kanalı 2 ses

AUD2DAT (0CA,W) : Ses kanalı 2 data

AUD3LCH (0D0,W) : Ses kanalı 3 DMA data'sı (üst 3 bit)

AUD3LCL (0D2,W) : Ses kanalı 3 DMA data'sı (alt 15 bit)

AUD3LEN (0D4,W) : Ses kanalı 3 uzunluk

AUD3PER (0D6,W) : Ses kanalı 3 periyod

AUD3VOL (0D8,W) : Ses kanalı 3 ses

AUD3DAT (0DA,W) : Ses kanalı 3 data

BPL1PTH (0DC,W) : Bit plane 1 pointer (üst 3 bit)

BPL1PTL (0DE,W) : Bit plane 1 pointer (alt 15 bit)

BPL2PTH (0E0,W) : Bit plane 2 pointer (üst 3 bit)

BPL2PTL (0E2,W) : Bit plane 2 pointer (alt 15 bit)

BPL3PTH (0E4,W) : Bit plane 3 pointer (üst 3 bit)

BPL3PTL (0E6,W) : Bit plane 3 pointer (alt 15 bit)

BPL4PTH (0E8,W) : Bit plane 4 pointer (üst 3 bit)

BPL4PTL (0EA,W) : Bit plane 4 pointer (alt 15 bit)

BPL5PTH (0EC,W) : Bit plane 5 pointer (üst 3 bit)

BPLSPTL (0EE,W) : Bit plane 5 pointer (alt 15 bit)

BPL6PTH (0F0,W) : Bit plane 6 pointer (üst 3 bit)

BPL6PTL (0F2,W) : Bit plane 6 pointer (alt 15 bit)

BPLCON0 (0FC,W) : Bit plane kontrol register'ı

BPLCON1 (0FE,W) : Bit plane kontrol reg. (kayma değeri)

BPLCON2 (100,W) : Bit plane kontrol reg. (öncelik kontrolü)

BPL1MOD (104,W) : Bit plane modulo (tek plane'ler)

BPL2MOD (106,W) : Bit plane modulo (çift plane'ler)

BPL1DAT (10C,W) : Bit plane 1 data (paralel->seri)

BPL2DAT (10E,W) : Bit plane 2 data (paralel->seri)

BPL3DAT (110,W) : Bit plane 3 data (paralel->seri)

BPL4DAT (112,W) : Bit plane 4 data (paralel->seri)

BPL5DAT (114,W) : Bit plane 5 data (paralel->seri)

BPL6DAT (116,W) : Bit plane 6 data (paralel->seri)

SPR0PTH (120,W) : Sprite 0 pointer (üst 3 bit)

SPR0PTL (122,W) : Sprite 0 pointer (alt 15 bit)

SPR1PTH (124,W) : Sprite 1 pointer (üst 3 bit)

SPR1PTL (126,W) : Sprite 1 pointer (alt 15 bit)

SPR2PTH (128,W) : Sprite 2 pointer (üst 3 bit)

SPR2PTL (12A,W) : Sprite 2 pointer (alt 15 bit)

SPR3PTH (12C,W) : Sprite 3 pointer (üst 3 bit)

SPR3PTL (12E,W) : Sprite 3 pointer (alt 15 bit)

SPR4PTH (130,W) : Sprite 4 pointer (üst 3 bit)

SPR4PTL (132,W) : Sprite 4 pointer (alt 15 bit)

SPR5PTH (134,W) : Sprite 5 pointer (üst 3 bit)

SPR5PTL (136,W) : Sprite 5 pointer (alt 15 bit)

SPR6PTH (138,W) : Sprite 6 pointer (üst 3 bit)

SPR6PTL (13A,W) : Sprite 6 pointer (alt 15 bit)

SPR7PTH (13C,W) : Sprite 7 pointer (üst 3 bit)

SPR7PTL (13E,W) : Sprite 7 pointer (alt 15 bit)

SPR0POS (140,W) : Sprite 0 dikey-yatay start pozisyon data

SPR0CTL (142,W) : Sprite 0 dikey bitiş pozisyonu ve kontrol register'ı

SPR0DATA (144,W) : Sprite 0 image data register'ı A

SPR0DATB (146,W) : Sprite 0 image data register'ı B

SPR1POS (148,W) : Sprite 1 dikey-yatay start pozisyon data

SPR1CTL (14A,W) : Sprite 1 dikey bitiş pozisyonu ve kontrol register'ı

SPR1DATA (14C,W) : Sprite 1 image data register'ı A

SPR1DATB (14E,W) : Sprite 1 image data register'ı B

SPR2POS (150,W) : Sprite 2 dikey-yatay start pozisyon data

SPR2CTL (152,W) : Sprite 2 dikey bitiş pozisyonu ve kontrol register'ı

SPR2DATA (154,W) : Sprite 2 image data register'ı A

SPR2DATB (156,W) : Sprite 2 image data register'ı B

SPR3POS (158,W) : Sprite 3 dikey-yatay start pozisyon data

SPR3CTL (15A,W) : Sprite 3 dikey bitiş pozisyonu ve kontrol register'ı

SPR3DATA (15C,W) : Sprite 3 image data register'ı A

SPR3DATB (15E,W) : Sprite 3 image data register'ı B

SPR4POS (160,W) : Sprite 4 dikey-yatay start pozisyon data

SPR4CTL (162,W) : Sprite 4 dikey bitiş pozisyonu ve kontrol register'ı

SPR4DATA (164,W) : Sprite 4 image data register'ı A

SPR4DATB (166,W) : Sprite 4 image data register'ı B

SPR5POS (168,W) : Sprite 5 dikey-yatay start pozisyon data

SPR5CTL (16A,W) : Sprite 5 dikey bitiş pozisyonu ve kontrol register'ı

SPR5DATA (16C,W) : Sprite 5 image data register'ı A

SPR5DATB (16E,W) : Sprite 5 image data register'ı B

SPR6POS (170,W) : Sprite 6 dikey-yatay start pozisyon data

SPR6CTL (172,W) : Sprite 6 dikey bitiş pozisyonu ve kontrol register'ı

SPR6DATA (174,W) : Sprite 6 image data register'ı A

SPR6DATB (176,W) : Sprite 6 image data register'ı B

SPR7POS (178,W) : Sprite 7 dikey-yatay start pozisyon data

SPR7CTL (17A,W) : Sprite 7 dikey bitiş pozisyonu ve kontrol register'ı

SPR7DATA (17C,W) : Sprite 7 image data register'ı A

SPR7DATB (17E,W) : Sprite 7 image data register'ı B

COLOR00 (180,W) : Renk değeri 00

COLOR01 (182,W) : Renk değeri 01

...

COLOR31 (1BE,W) : Renk değeri 31

Bu register'lara şimdilik sadece göz atmanız yeterli. Birçok konuda size fikir verecektir. İleride blitter, copper, ses ve görüntü alanları ile ilgili bu register'lara çokça ihtiyaç duyacağız.

# Örnek Program 1 – ASCII2Hex Converter

Aşağıdaki örnek kod A0'da adresini verdiğiniz 8 karakterlik ifadeyi okuyup D0'da hexadecimal değerini döndürür. Açıklamalar kodun içindedir.

```
start:
        ;try adresini A0'a koy
        ;tohex isimli altrutini cagir
        lea         try(pc),a0
        jsr         tohex
        rts

;
; Ascii -> Hex Cevirici
; A0: string baslangic adresi
; Sonuc: D0 (longword)
; Degisen register'lar: D0/D1/D2/A0
;
tohex:
        ;D0 = 0, D1 = 0, D2 = 0
        clr.l         d0
        clr.l         d1
        moveq         #0,d2
.loop:
        ;A0'in gosterdigi yerden
        ;1 byte'i D0'a oku
        move.b         (a0)+,d0

        ;hexit altrutinini cagir
        bsr.s         .hexit

        ;D2'nin degerini 1 arttir
        ;8 olmadiysa loop'a git
        addq         #1,d2
        cmp         #8,d2
        bne.s         .loop

        ;sonucu D1'den D0'a kopyala
        ;ve programdan cik
        move.l         d1,d0
        rts

.hexit:
        ;D0'daki deger 'a' ($61)'den kucukse
        ;nolowercase'e git, deger a-f degildir
        ;aksi halde 'a'-10 ($51) cikart
        ;boylece deger 'a' ise $0A kalir
        ;ve out'a dallan
        cmp.b         #'a',d0
        blt.s         .nolowercase
        sub.b         #'a'-10,d0
        bra.s         .out

.nolowercase:
        ;D0'daki deger 'A' ($41)'den kucukse
        ;nouppercase'e git, deger A-F degildir
        ;aksi halde 'A'-10 ($31) cikart
        ;boylece deger 'A' ise $0A kalir
        ;ve out'a dallan
        cmp.b         #'A',d0
        blt.s         .nouppercase
        sub.b         #'A'-10,d0
        bra.s         .out

.nouppercase:
        ;D0'daki deger '0'-'9' arasi ($30-$39)
        ;'0' cikart ve byte degerini elde et
        sub.b         #'0',d0
.out:
        ;sonucu tutacak olan D1 registerini
        ;4 bit sola kaydir (1 nibble, yarim byte)
        ;ve elde edilen D0'daki deger ile OR'la
        ;sonuc yine D1'de durur
        lsl.l         #4,d1
        or.b         d0,d1
        rts

try:
        dc.b         "3a06DfC0"
```

# Örnek Program 2 – Copper Bar

Aşağıdaki kod örnek bir copper listesidir. Henüz bu konulara girmemiş olmamıza rağmen hardware register'ları ile ilgili olduğundan burada yer verdim (biraz da ilginizi çekmek için), önümüzdeki sayıda copper ve görüntü alanları (display fields) hakkında daha detaylı bilgi bulabilirsiniz. Şimdilik sadece yazıp inceleyiniz.

```
;
; Ornek Copper List
;

;bazi hardware register ofsetleri
bplcon0 = $100
bplcon1 = $102
bpl1mod = $108
diwstrt = $08E
diwstop = $090
ddfstrt = $092
ddfstop = $094
color00 = $180
color01 = $182
cop1lc  = $080
copjmp1 = $088
dmacon  = $096
bpl1pth = $0E0
```

```
bpl1ptl = $0E2

start:
        ;Custom Chip taban adresi
        lea     $dff000,a0
        ;1 bitplane (tek renk)
        move.w  #$1200,bplcon0(a0)
        move.w  #0,bplcon1(a0)
        move.w  #0,bpl1mod(a0)
        ;goruntunun baslangic/bitisi
        move.w  #$0038,ddfstrt(a0)
        move.w  #$00d0,ddfstop(a0)
        move.w  #$2c81,diwstrt(a0)
        move.w  #$f4c1,diwstop(a0)
        ;bitplane adresini copper
        ;liste yerlestir
        move.l  #bitplane,d0
        move.w  d0,cl
        swap    d0
        move.w  d0,ch
        ;copper listesinin baslangic
        ;adresini yaz
        move.l  #copperlist,cop1lc(a0)
        ;copper'i tetikle
        move.w  copjmp1(a0),d0
        ;dma'lari ayarlar
        move.w  #$8380,dmacon(a0)
        ;sol tusa basilana kadar bekle
.leftclick:
        btst    #6,$bfe001
        bne.s   .leftclick
        rts

        ;copper list ve bitmap daima
        ;CHIP Ram'de durmalidir

        section data,data_c

copperlist:
        dc.w    bpl1pth
ch:   dc.w    0
        dc.w    bpl1ptl
cl:   dc.w    0
        ;sprite'i kapat
        dc.w    $0120,$0000
        dc.w    $0122,$0000
        ;$96 raster satirini bekle
        dc.w    $9601,$ff00
        ;color00'a (zemin rengi,$0180)
        ;RGB $678 degeri gir
        dc.w    $0180,$0678
        dc.w    $9701,$ff00
        dc.w    $0180,$0789
        dc.w    $9801,$ff00
        dc.w    $0180,$089a
        dc.w    $9901,$ff00
        dc.w    $0180,$09ab
        dc.w    $9a01,$ff00
        dc.w    $0180,$0567
        ;copper listesini bitir
        dc.w    $ffff,$fffe

;Bitplane icin bos alan ayir

bitplane:
        ds.b    320*256
```

Yukarıdaki örnek kodda Bitmap için gerekli boş hafıza alanını kodun içinde ayırdık. Elbette birçok durumda bunun için Exec library'nin hafıza ayırma fonksiyonlarını (memory allocation functions, AllocMem) kullanmamız gerekir. Ancak bir süre daha örnek kodlarımızı bu şekilde yapmaya devam edeceğiz. Library'ler konusuna ise daha ileriki sayılarda gireceğiz.

Bu sayıdaki yazımızın sonuna geldik. Bu sayıda dallanma komutlarını öğrendik ve hardware registerler'ı hakkında bilgi edindik, örnek kodlarla da öğrendiklerimizi uygulamış olduk. Ayrıca dergi ile birlikte AsmPro'yu doğrudan çalıştırabileceğiniz bir ADF dosyasını bulabilirsiniz. Buradaki örnek kodları disketin içindeki sources klasöründe bulabilirsiniz.

Bir sonraki sayıda görüşmek üzere hosçakalın.

Kaynaklar:


1. Amiga Machine Language – Abacus
2. Amiga Hardware Reference Manual
3. Programming the 68000 - Steve Williams. (c) 1985 Sybex Inc

# AsmPro Kılavuzu

## Şemseddin 'Endo' Moldibi

Merhaba Amiga'da assembly dilini kullanarak kod yazmak için kullanabileceğiniz pek çok assembler vardır. Bunların en tanınmış olanları Master Seka, Trash'm One, PhxAss, AsmOne ve AsmPro'dur. Bunlar dışında Aminet'te pek çok güzel derleyici bulabilirsiniz.

Bu yazımızda AsmPro ve AsmOne'yu nasıl kullanacağımızı öğreneceğiz. Bu iki program birbirinden bağımsız olsalar da özellikleri çok benzerdir.

Size tavsiyem artık Master Seka ve Trash'm One kullanmamanızdır. Çok uzun yıllardır geliştirilmiyorlar ve özellikle yeni sistemlerde problemlidirler. AsmPro'nun AsmOne'a göre en önemli avantajı ise grafik kartlı amigalarda, grafik kartının ekran modlarını kullanabilmesidir. Ancak AsmOne'ın son versiyonlarında bu özellik de eklendi.

PhxAss ise çok kaliteli ve güncel olmasına karşın bir IDE'ye sahip değildir. Elbette CED ya da GoldEd gibi bir text editörü ile kullanabilirsiniz. Ancak yine de bir debugger kullanmanız gerekir. Bu konuda da Barfly'ı tavsiye ederim.

Yazının geri kalanını AsmOne üzerinden anlatmaya devam edeceğim, çok büyük ölçüde AsmPro için de aynı olacaktır.

AsmOne bir; Editör, Assembler, Debugger ve Monitör programıdır. Yani kodlarınızı yazabilir, derleyebilir, satır satır çalıştırarak ya da breakpoint koyarak debug edebilirsiniz. Ayrıca hafızanın herhangi bir bölümünü görüntüleyebilir ve disassemble edebilirsiniz. AsmOne ile aynı anda 10 farklı kaynak dosyası üzerinde çalışabilir ve bunları bir proje olarak kaydedebilirsiniz.

# AsmOne/Pro Komutlar / Menüleri

AsmOne ile çalışırken kullanabileceğiniz 2 yöntem vardır. Bunlardan biri programın menülerini kullanmak diğeri ise (kendi dokümanında DLC = Direct Line Command olarak isimlendirdiği) çoğunlukla tek karakterlik komutları komut satırına yazarak kullanabilirsiniz.

Dilerseniz önce komut satırından verilen komutların genel bir listesine bakalım:

Komutların bazıları parametre almazken bazıları birden fazla parametre alır, o durumda karşınıza prompt çıkacaktır. O komutlar için parametre bölümünde "?" göreceksiniz.

## AsmOne Komutları

| Komut | Parametre | Açıklama |
|---|---|---|
| | | |
| ! | - | Çık/İlk ekrana dön |
| !! | - | Hemen çık |
| # | - | Hakkında penceresi |
| = | - | Hafıza kullanım bilgisi |
| =C | - | Program renklerini değiştir |
| =F | - | Font'u değiştir |
| =M | - | Projeye hafıza ekle |
| =P | - | Proje bilgisi |
| =R | - | Register bilgisi |
| =S | - | Sembol listesi (label vs.) |
| > | dosya ismi | Çıktıyı dosyaya yönlendir |
| ? | value | Hex/Ascii/Bin çevirici |
| [ | value | Ondalıklı işlem |
| @A | adres/label | Hafıza editleme |
| @B | adres/label | Hafızayı binary göster |
| @D | adres/label | Hafızayı disassemble et |
| @H | adres/label | Hafızayı hexadecimal listele |
| @N | adres/label | Hafızayı ascii listele |
| A | - | Kodu derle |
| AC | - | Kodu test et |
| AD | - | Derle ve debugger aç |
| AO | - | Assemble ve optimize et |
| B | - | Son satıra git (Editörü açar) |

| | | |
|---|---|---|
| C | ? | Hafızanın bir bölümünü kopyala |
| CC | sürücü | Disketin Checksum'ını hesapla |
| CS | ? | Sinüs tablosu oluştur |
| D | - | Disassembler'ı aç |
| E | dosya ismi | Harici dosya yükle |
| EL | ? | Tüm labelların başına/sonuna verilen ifadeyi ekle |
| F | ? | Hafızayı doldur |
| G | | Adrese git (Go) |
| H | adres/label | Monitere geç (hexadecimal) |
| I | dosya ismi | Dosyayı araya sok (Insert) |
| IB | ? | Hafızayı binary olarak araya sok |
| ID | ? | Hafızayı disassembly ederek araya sok |
| IH | ? | Hafızayı hex olarak araya sok |
| IN | ? | Hafızayı ascii olarak araya sok |
| IS | ? | Sinüs tablosu sok |
| J | adres | Adrese git (Jump) |
| K | adım sayısı | Adım adım çalıştır |
| L | ifade | Verilen yazıyı bul |
| M | - | Monitör'e geç |
| N | - | Monitör'e geç (Ascii) |
| O | - | Silinmiş kodu geri getir (Old) |

| | | |
|---|---|---|
| P | satır sayısı | Yazdır (bulunduğu pozisyondan) |
| PS | ? | Debug işlemi için parametre |
| Q | ? | İki hafıza bölümünü karşılaştır |
| R | dosya ismi | Dosya aç |
| RB | dosya ismi | Binary dosya aç |
| RE | dosya ismi | Proje aç |
| RO | dosya ismi | Obje (exe) dosyası aç |
| RS | sürücü, ? | Disketten sektör oku |
| RT | sürücü, ? | Disketten track oku |
| S | ? | Hafızada bul |
| T | - | İlk satıra git (Editörü aç) |
| U | - | Dosyayı güncelle |
| UA | - | Tüm dosyaları güncelle |
| V | klasör | Klasörü listeler (Dir) |
| W | dosya ismi | Dosyaya yaz |
| WB | dosya ismi | Binary dosya yaz |
| WE | dosya ismi | Projeyi kaydet (Tüm açık dosyaları) |
| WL | dosya ismi | Link dosyası kaydet |
| WN | dosya ismi | Kodu ascii olarak |
| WO | dosya ismi | Obje (exe) oluştur |
| WP | - | Ayarları kaydet |
| WS | ? | Sektör yaz |
| WT | ? | Track yaz |
| X | -/register | Registerları listele |

| | | |
|---|---|---|
| Y | komut | DOS komutu çalıştır |
| ZA | - | Derlemede kullanılan hafızayı serbest bırak |
| ZB | - | Breakpoint'leri temizle |
| ZF | | Dosyayı sil (disketten) |
| ZI | - | Include'ları temizle, derleme işleminde tekrar yüklenirler |
| ZS | - | Üzerinde çalışılan dosyayı sil |
| CD | klasör adı | Klasör yarat |

Bu listedeki pek çok komut yazılıp enter'a basıldıktan sonra parametrelerini almaktadır, bu komutların parametrelerini "?" ile gösterdim. Örneğin;

> C

> BEG> $00234500

> END> $00302500

# AsmOne Menüleri

AsmOne'ın menülerinden ulaşabileceğiniz tüm komutlar aslında prompt'tan ulaştıklarınızın aynısıdır. Bu nedenle tekrar açıklamalarını yapmaya gerek görmüyorum. Tüm menü seçeneklerinin yanında o işlemi gerçekleştirmek için kullanabileceğiniz komutu görebilirsiniz. Editör ve Debugger içindeki menü seçenekleri de isimlerinden kolayca anlaşılabilecek seçeneklerdir. Editörde blok komutları, kod içinde belirli satırlara atlama; Debugger'da ise breakpoint ve watch ekleme, satır satır çalıştırma gibi özellikler vardır.

# AsmOne Ayarları

AsmOne/Pro ayarları iki bölümdedir: Assembler Preferences ve Environment Preferences.

Assembler Preferences, derleme işlemi ile ilgili ayarlardır;

| | |
|---|---|
| Rescue | Kaynak kodunuzun ZS ile silindikten sonra O (Old) komutu ile geri alınabilmesini sağlar. |
| Level 7 | AsmOne Level 7 interrupt kullanır. |
| NumLock | Açık olması durumunda nümerik klavyeyi yön tuşları olarak kullanabilirsiniz. |
| Auto Alloc | Section'lar için hafıza ayırma işlemini otomatik yapar. |
| Debug | Debugger geçmeden önce kodun otomatik derlenip derlenmeyeceğini belirler. |
| List File | Derleme işlemi sırasında kod listesini gösterir; satır numaraları, label'lar vb. |
| Paging | Listelemeyi sayfa sayfa yapar. |
| All Errors | Hata ile karşılaşıldığında derleme işlemi devam eder. Birçok hatayı aynı anda görmek için kullanabilirsiniz. |
| Progress Indicator | Derleme işleminde progress gösterir. |
| Progress By Line | Derlenmekte olan satırı gösterir. |
| DS Clear | DS ile ayrılan bloğun temizlenmesini sağlar. Normalde bu alan herhangi bir değer ile doldurulmaz. Derlenmiş programda temizlenmeyeceğini unutmayın. |
| Label : | Label'lar : ile bitmesi gerektiğini gösterir. |
| UCase = LCase | Büyük/Küçük harfin eşit kabul edilip edilmeyeceği. |
| ; Comment | Açıklama satırlarının ; ile başlayıp başlamayacağı. |
| Processor Warn | Kullandığınız komutların seçili CPU tarafından desteklenmemesi durumunda uyarı gösterilir. |
| FPU Present | Sisteminizde FPU olup olmadığını seçebilirsiniz. |
| 68020++ Odd Data | 68020 ve üstü işlemcilerde data'larınız tek adreslere denk gelebilir. 68000'de bu durumda |

| | |
|---|---|
| | hata oluşur. Tek adrese denk gelen data olması durumunda uyarı verilip verilmeyeceğini bu seçenek ile belirleyebilirsiniz. |
| 68851 Present | Sisteminizde MMU olup olmadığını seçebilirsiniz. |

Environment Preferences, uygulama ayarlarınızın bazılarının açıklamaları aşağıdaki gibidir:

| | |
|---|---|
| ReqTools Library | Dosya açma / kaydetme işlemleri için ReqTools library kullanılır. |
| Save Marks | Dosyanızın içine belirlediğiniz mark'lar da kaydedilir. |
| WB to front | Kodunuzu çalıştırdığınızda Workbench öne getirilir. |
| Resident Registers | REGSDATA dosyası hafızada bekletilir. =R ile tüm donanım register'larının listesini görebilirsiniz. |
| Safety | Popup pencerelerde varsayılan seçenek olmaz. |
| Close Workbench | AsmOne açıldığında WB kapatılır. WB'den çalışan başka programlar varsa kapatılamayabilir. |
| Clipboard Support | Clipboard açık/kapalı. |
| Parameters | Debug işlemi başlatılırken koda parametre verilip verilemeyeceği. PS ile verilecek parametreleri belirleyebilirsiniz. |
| RTG Mode | Grafik kartı desteği açık/kapalı. DİKKAT! Eğer AGA ekranda çalışıyorsanız ve bu seçenek seçiliyse Debugger'a girerken hata alabilirsiniz. Grafik kartı ekranında çalışmıyorsanız seçmeyiniz. |
| Extended ReqTools | Bul/değiştir gibi işlemler için ReqTools library'i kullan/ kullanma. |
| Keep x | Editörde aşağı/yukarı ilerlerken imleç aynı kolon hizasında kalır. |
| ASCII Only | Monitörde ASCII olmayan karakterler gösterilmez. |
| Disassembly | Debugger'da bir sonraki satırı gösterir. |
| Show Source | Debugger ekranında derlenmiş kod ya da kaynak kod gösterilmesini sağlar. |
| Enable/Permit | Kod çalıştırılmadan önce multitasking'in kapatılıp kapatılmayacağı. |
| Libcalls dec | Library fonksiyonları için ofset değerleri decimal moda gösterilir. |

# AsmOne Direktifleri

Aşağıdaki kodlarınızın içinde kullanabileceğiniz direktifleri ve açıklamalarını bulabilirsiniz.

| | | |
|---|---|---|
| EXTERN | [label] EXTERN [number,]<file>,<address>[,length] | Harici bir dosyayı verilen adresten itibaren yükle |
| = | <label> = <value> | label'a <değer> atar. |
| * | <label> *<operator> | Derleme işleminde o anki hafıza adresini gösterir. |
| ADDWATCH | ADDWATCH <label> | Debugger için bir izleme yeri ekler. |
| ALIGN | ALIGN <value1>,<value2> | Sonraki satırın denk geleceği adresi belirler. Değer2'ye bölünebilen, sonraki ilk adres bulunur ve değer1 eklenir, derleme işlemine bu adresten itibaren devam edilir. Ör: ALIGN 0,4 bir sonraki 4'e bölünebilen (longword) adrese geç. Burada arada atlanan byte'lara herhangi bir değer yazılmaz. |
| AUTO | AUTO <command>[\comm | Derleme sırasında diğer DLC'lerin çal- |

| | | |
|---|---|---|
| | and..] | ıştırılmasını sağlar, komutlar \ ile ayrılır. |
| BASEREG | BASEREG <label>,<register> | Verilen adres reg. için bir ofset değeri atar, sonraki tüm relatif işlemlerde bu ofset değeri otomatik olarak eklenir. |
| BLK | [label] BLK.[B\|W\|L\|D\|S] <value1>,<value2> | Sabit değerlerden oluşan bir blok tanımlar. Ör: BLK.W 100,$AA55 ;100 adet $AA55 (word) |
| CMEXIT | CMEXIT <value> | İç içe çağırılan makrolardan <değer>'e ulaşıldığında çıkılmasını sağlar. |
| CNOP | CNOP <value1>,<value2> | Bkz. ALIGN |
| DC | DC.[size] <expresion>[,expresion...] | Sabit veri tanımlamanızı sağlar, Ör: DC.B $01,$02,'plazma',$80 / 2 |
| DCB | DCB.[size] <value1>,<value2> | Bkz. BLK |
| DR | DR.[B\|W\|L] <value> | Relatif değerlerden oluşan bir blok tanımlamanızı sağlar. Bu değerler bulunduğu adres ile verilen değer arasındaki fark ile hesaplanır (<value>-*). Ör: DR.L myRoutine |
| DS | DS.[size] <value> | İleride kullanılmak üzere <değer> kadar alan ayırır. Ör: DS.W 50 ;50 word (100 byte) alan ayır. Dikkat! Bu ayrılan alan herhangi bir değer ile doldurulmaz. |
| ELSE | | IF ile kullanılır |
| END | | Kodun bittiği yeri gösterir, o noktadan sonrası derlenmez. |
| ENDB | ENDB <address register> | BASEREG tanımlamasını kaldırır. |
| ENDC | | ENDIF ile aynıdır. |
| ENDIF | | IF bloğunu bitirir. |
| ENDM | | Makro bloğunu bitirir. |
| ENDOFF | | OFFSET tanımlamasını kaldırır. |
| ENDR | | REPT bloğunu bitirir. |
| ENTRY | | Harici bir tanımlama yapar. Bkz. XDEF, EXTRN, GLOBAL. |
| EQU | <label> EQU <value> | <label>a <değer> atar. |
| EQUC | <label> EQUC <PPC register> | PPC reg.'na <isim> verir |
| EQUD | <label> EQUD <FPU value> | <label>a <değer> atar. (Double, Ondalık) |
| EQUP | <label> EQUP <FPU value> | <label>a <değer> atar. (Paxked) |
| EQUR | <label> EQUR <register> | <register>a <isim> verir. |
| EQUS | <label> EQUS <FPU value> | <label>a <değer> atar. (Single, Ondalık) |
| EQUX | <label> EQUX <FPU value> | <label>a <değer> atar. (Extended) |
| EREM | | REM bloğunu bitirir |
| ETEXT | | TEXT bloğunu bitirir |
| EVEN | | Sonraki adresin bir çift (even) adres olmasını sağlar (CNOP 0,2) |
| EXTRN | EXTRN <label>[,label...] | Harici tanımala yapar. C'deki extern ile aynıdır. |
| | | |

| | | |
|---|---|---|
| FAIL | | 'User made FAIL' hatası verilmesini sağlar. |
| FILESIZE | FILESIZE(<file>) | Harici bir dosyanın boyutunu okur. Özellikle derleme sırasında Memory Allocation için kullanışlıdır. |
| GLOBAL | | Bkz. EXTRN |
| IDNT | IDNT <string> | ??? |
| IF | IF(EQ\|NE\|GT\|GE\|LT\|LE) <bool-value> | IF bloğu açar |
| IF1 | | Bu blok içindekiler sadece derlemenin ilk aşamasında derlenir. (AsmOne/Pro 2 aşamalı (2 Pass) bir derleyicidir) |
| IF2 | | Bu blok içindekiler sadece derlemenin ikinci aşamasında derlenir. |
| IFB | | ??? |
| IFC | IFC <string1>,<string2> | <string1> = <string2> olduğunda IF bloğu derlenir. |
| IFD | IFD <symbol> | <sembol> tanımlı ise blok derlenir. |
| IFNB | | ??? |
| IFNC | IFNC <symbol1>,<symbol2> | <string1> ile <string2> eşit olmadığında blok derlenir. |
| IFND | IFND <symbol> | <sembol> tanımlı değil ise blok derlenir. |
| IMAGE | IMAGE <file>[,<address>] | Bkz. INCBIN |
| INCBIN | INCBIN <file>[,<address>] | Binary dosyayı hafızaya okur. Adres verilirse verilen adrese yükler. |
| INCDIR | INCDIR <path> | INCBIN vb. komutlar için path belirtir. Ör: INCIDR "HD0:sources/" ;sondaki / karakterini unutmayın. |
| INCIFF | | IFF dosya include eder |
| INCIFFP | | IFF palet include eder |
| INCLUDE | INCLUDE <file> | Verilen kaynak kod dosyasını include eder. Bu işlem sadece ilk derlemede bir defa yapılır, böylece derleme işleminde hız sağlanır. ZI ile hafızaya yüklenmiş olan bu dosyaları temizleyip yeniden yüklenmelerini sağlayabilirsiniz. |
| INCSRC | INCSRC <sourcenumber> | <sourcenumber> numaralı kaynak kodu include edilir. |
| JUMPERR | JUMPERR <label> | Kendi hata rutinizine atlamanızı sağlar |
| LOAD | LOAD <address> | Derleme işleminin belirli bir adresten itibaren yapılmasını sağlar. Bkz. ORG |
| MACRO | | Makro bloğu açar. |
| MEXIT | | Makro tanımlamasından çıkar. Tanımlamayı bitirmek için ENDM kullanın. |
| ODD | | Sonraki adresi tek (odd) adres yapar. |
| OFFSET | | |
| ORG | ORG <address> | Program için bir mutlak (absolute) başlangıç adresi belirler. LOAD ile birlikte programın- |

| | | |
|---|---|---|
| | | ızın belirli bir hafıza adresine yerleştirilmesini sağlayabilirsiniz. |
| PRINTT | PRINTT <string> | Verilen ifadeyi ekrana yazar. |
| PRINTV | PRINTV <label/value>[,label/value...] | Verilen <label> ya da <value>'ları ekrana yazar. |
| REM | | Açıklama bloğu açar. REM / EREM arasında kalan yazılar derlenmez. ; ile tek satırlık açıklamalar girebilirsiniz. |
| REPT | REPT <number> | REPT / ENDR arasındaki blok <number> kadar tekrarlanır. |
| RS | RS.[size] <value> | Dâhili RS sayacına <value> ekler. Örneklere bakınız. |
| RSRESET | | RS sayacını sıfırlar. |
| RSSET | RSSET <value> | RS sayacını verilen değere eşitler. |
| SECTION | [label] SECTION <name>[,type][_memory] | Yeni bir program bölümü (section) başlatır. Tip CODE, DATA ve BSS olabilir, hafıza için _C (chip), _F (fast) and _P (public) kullanılabilir. Copper list, ses ve görüntü verilerini CHIP RAM'e koymak için kullanabilirsiniz. |
| SET | [label] SET <value> | [label]'ı <value>'ya eşitler. EQU ve = ile aynıdır, ancak daha sonra aynı label için farklı değer SET edilebilir. |
| SETCPU | SETCPU <option> | CPU tipini belirler: 000, 010, 020, 030, 040, 060 olabilir. |
| SETFPU | SETFPU <option> | FPU olup olmadığını belirler: ON, OFF verilebilir. |
| SETMMU | SETMMU <option> | MMU olup olmadığını belirler: ON, OFF verilebilir. |
| TEXT | | TEXT bloğu başlatır. Bkz. ETEXT |
| XDEF | XDEF <label>[,label....] | Diğer kodlarda kullanılmak üzere verilen label'ları harici olarak tanımlar. |
| XREF | XREF <label>[,label....] | Verilen label'ların harici tanımlar olduğunu belirtir. Derleme işleminde aktif kaynak kodda bu ifadeler tanımlı olmasa da derleme işlemi devam eder. |

# Hata Mesajları

| Hata Mesajı | Açıklama |
|---|---|
| Workspace Memory Full | Proje için ayrılan hafıza bitti, =M kullanın |
| Address Register Byte/Logic | Adres reg.'larına .B ile ulaşamazsınız.<br>Ayrıca lojik işlem uygulanamaz.<br>Ör. OR.W A0,D0 |
| Address Register Expected | Adres reg. gereken yerde data reg. kullanıldı |
| Comma Expected | , ile ayrılmış argüman gerekli |
| Data Register Expected | Data reg. gereken yerde adres reg. kullanıldı |
| Double Symbol | Aynı ifade iki kez tanımlanmış |
| Unexpected End of File | Bir MACRO/IF ifadesi açık kalmış |
| End of File | Dosya sonuna ulaşıldı (Search) |
| User made FAIL | FAIL komutuna rastlandı |

| | |
|---|---|
| | (compile-time) |
| Illegal Command | Geçersiz komut |
| Illegal Address Size | Geçersiz adres boyutu. Ör. CLR.B A0 |
| Illegal Operand | Geçersiz argüman |
| Illegal Operator | Geçersiz operatör |
| Illegal Operator in BSS Area | BSS içinde yalnızca DS kullanılabilir |
| Illegal Order | Argüman sırası hatalı. Ör: MOVEM.L A0,D4-D0.-(A7) ;Önce data sonra adres reg. verilmelidir. |
| Illegal Register Size | Geçersiz reg. boyutu. Ör: MOVE.L (A0,D0.B),D1 ;Index reg. .W ya da .L olabilir |
| Illegal Section Type | Geçersiz SECTION tipi. Geçerli tipler: code, data, bss |
| Illegal Size | Hatalı boyut. Ör: ADDA.B D0,A0 ;adres reg. ile .B işlem yapılamaz |
| Illegal MACRO Definition | Geçersiz MACRO tanımı |
| Immediate Operand Expected | Bazı komutlar sadece |
| Include Jam | Include hatası |
| MACRO Overflow | İç içe en fazla 25 macro tanımlanabilir |
| Invalid Address Mode | Geçersiz adres modu |
| LOAD without ORG | LOAD komutu yalnıza ORG ile kullanılabilir |

# Makro Tanımlama

AsmOne/Pro bir makro assembler'dır. Yani kaynak kodlarınız içinde sık kullandığınız işlemleri kolayca yapmak ve tekrar tekrar yazmamak için çeşitli makrolar tanımlayabilirsiniz. Bu bölümde nasıl makro tanımlayacağımızı göreceğiz.

Bir makro tanımı yapmak için en uygun yer kodunuzun ilk satırlarıdır. Ayrıca tüm makrolarınızı ayrı bir include dosyasına da koyabilirsiniz. Bir makro tanımlaması aşağıdaki şekilde yapılır:

```
MYMACRO: MACRO
         ...
         ENDM
```

Burada MYMACRO tanımladığımız makronun ismidir. İsmin büyük harfle yazılması şart değildir ancak kod içinde makro çağırılan yerleri kolayca görmek açısından bu şekilde kullanmak iyidir. Makro ismi aynı zamanda bir label gibidir bu nedenle ayarlarınızda label sonlarında : olması gerektiğini seçtiyseniz burada da olması gerekir. Ayrıca isim ile MACRO ifadesi aynı satırda olmalıdır.

```
CLEAN: MACRO
           moveq #0,d0
           moveq #0,d1
           moveq #0,d2
       ENDM
```

## Makrolara Argüman Verme

Tanımladığınız makrolara 10 taneye kadar argüman ekleyebilirsiniz. Bunun için \0 - \9 ifadelerini kullanabilirsiniz. Aşağıdaki örneği inceleyin:

```
CLEAN: MACRO
           moveq #0,\1
       ENDM

TEST:  MACRO
           move.l (\1)+,(\2)+
       ENDM

       CLEAN D3
       TEST  A0,A1
```

Ayrıca NARG isimli değişken makro içinde, makronun kaç argümanla çağırıldığını görmek ve buna göre farklı adımlar uygulamak için kullanılabilir:

```
CLEAN: MACRO
                IF NARG = 2
                        clr.l \1
                        clr.l \2
                ELSE
                        FAIL
                ENDIF
         ENDM
```

Daha karmaşık bir makro örneği aşağıdadır (AsmOne 1.4 Manual'den alıntıdır):

```
do_it:  MACRO
            btst        #\1,d0
            bne.b        \@1
            move.l        \2,d0
            lsl.w        #\3,d0
            bra.b        *+4
\@1:              moveq         #0,d0
        ENDM

start:
        do_it        10,d2,3
```

Include dosyalarından (özellikle exec/types.i) pek çok makronun

nasıl tanımlandığını görebilirsiniz.

### Makrolar Hakkında Dikkat Edilmesi Gerekenler

AsmOne/Pro'da makro yazarken bazı kurallara dikkat etmemeniz durumunda bulunması çok güç hatalarla karşılaşabilirsiniz. Çünkü AsmOne, hataları makronun içinde değil makronun çağırıldığı yerde verecektir. Bu yüzden aşağıdaki konulara dikkat edin;

1. Açıklama (comment) satırlarını makroların dışına yazın.
2. Makro içinde \ karakterini argümanlar dışında bir amaç için kullanmayın.
3. Makro tanımları içinde gereksiz hiçbir karakter kullanmayın.

## Include Dosyaları ile Çalışmak

Include dosyaları C'den bildiğimiz include dosyaları ile aynı işi görürler. ".i" uzantılıdırlar ve içerikleri AsmOne ile derlenebilecek şekildedir. İçlerinde çok çeşitli tanımlar tipler (types), yapılar (structure), makro tanımları ve sabitler (constants) vardır. Commodore firmasının yayınladığı "Includes & Autodocs" dosyalarını bulup hard diskinize ve bir diskete kopyalayabilir ve AsmOne'ın içinden yüklenerek kullanılabilirsiniz. Autodoc'lar tüm kütüphanelerin tüm fonksiyonlarını açıklayan yardım dosyalarıdır.

Ayrıca bu dosyalara ek olarak kütüphane ofset değerlerini barındıran LVO (Library Offsets) dosyaları vardır. Bu dosyalar çoğunlukla include klasörünün içinde LVO isimli ayrı bir klasörde dururlar. Bu dosyaları include ettiğinizde fonksiyonlar için ofset değerlerini ezbere bilmenize gerek kalmaz. Aşağıdaki örneği inceleyin:

```
incdir  "include:"
include "exec/types.i"
include "lvo/exec_lib.i"

CALL:   MACRO
            jsr    _LVO\1(a6)
        ENDM

;
; Open Gfx Library
;
        move.l $4.w,a6
        lea    gfxname(pc),a1
        CALL   OpenLibrary
        move.l d0,gfxbase
        beq.s NoGfx
```

Burada önce gerekli dosyalar include edilmiş, fonksiyon çağırma işlemini kolaylaştırmak için bir makro tanımlanmış ve OpenLibrary fonksiyonu çağırılmıştır. exec_lib.i dosyası include edilmemiş olsaydı CALL OpenLibrary yerine jsr -552(a6) yazmamız gerekecekti.

"types.i" dosyası ise tüm ön tanımlı ifadeleri barındırır. Herhangi başka bir dosyayı include etmeden önce bu dosyanın include edilmesi gerekir.

Ayrıca bu dosyaların içine bakarak da çok şey öğrenebilirsiniz. İçlerinde pek çok yararlı açıklama vardır.

Incdir "include:" ifadesinde tam yolu vermek yerine daha önceden yapılmış bir assign kullanılmıştır. CLI penceresinde assign komutu vererek assign'ların listesini alabilir ve yenilerini ekleyebilirsiniz. Ya da dilerseniz include dosyalarının tam yolunu yazabilirsiniz. INCDIR ifadesini kod içinde herhangi bir yerde tekrar vererek farklı klasörlerle çalışabilirsiniz.

Ayrıca belirtmekte fayda var; pek çok include dosyasının derlenebilmesi için ayarlarınızda UCase = LCase'in kaldırılması gerekir. Bazı durumlarda label'lar için : zorunluluğunu da kaldırmanız gerekebilir.

# Sonsözler

AsmOne/Pro kullanımı hakkında uzunca bir yazı oldu. Son olarak şunları ekleyebiliriz; AsmOne/Pro çok kullanışlı editörler olsa da bug'ları vardır, o nedenle çalışırken kodlarınızı hard disk ya da floppy diske kaydetmenizi öneririm. Benim yaptığım gibi RAM diske kaydetmeyin, aksi halde makineniz kilitlendiğinde kodlarınızı kurtaramayabilirsiniz.

Kodlarınızı yazarken her zaman label'ları en sol sütuna, komutları ise en az 1 tab karakteri ileriye yazın. Bu sadece okunabilirlik için değil sebebini anlamadığınız birçok hata mesajını da almanızı engelleyecektir.

Sonraki sayıya kadar hoşça kalın.

# CoZe'nin Amiga Donanım Köşesi

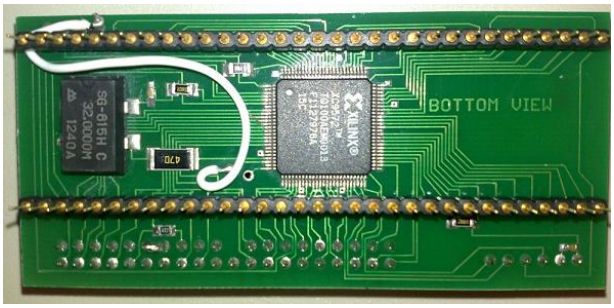
Arda (CoZe) Karaduman

Merhaba arkadaşlar, bu sayıdan itibaren bu köşede sizlere yeni/eski Amiga donanımları hakkında bilgi vermeye çalışacağım. Bu sayıdaki temamız, Amiga kullanıcılarının çabalarıyla gerçekleştirilen 'Home Brew' olarak tabir edebileceğimiz projeler. Gelecek sayılarda duruma göre profesyonel ürünler, ev tasarımları, eskiden üretilmiş ama fazla tanınmayan ürünler, ve çok kullanılan Amiga donanımları için püf noktaları gibi konuları köşemizde ele alacağız.

## 68000 Soket için IDE portu

68000 Soket için IDE portu

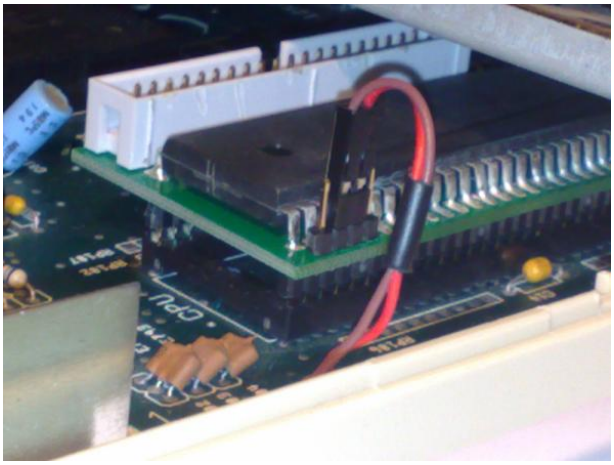
Bu proje Finlandiya'dan Mika Leinonen tarafından geliştirilen bir proje. Temel olarak Amiga 600/1200 lerde kullanılan Gayle çipinin emulasyonuna dayanıyor. IDE portu bulunmayan ve IDE arabirimi edinmenin oldukça zor olduğu Amiga 500, 2000 gibi 68000 bazlı Amiga kullanıcılarına yönelik bir proje. Kickstart 37.350 ve sonrası Gayle'in IDE birimi için sürücü içerdiğinden bu kartı scsi.device içeren tüm kickstartlarda kullanabilirsiniz*. Temel olarak Amiga 500 için tasarlanmış olsada, 68000 soket barındıran tüm sistemlerde (Amiga 2000, 1000, CDTV) çalışması muhtemel.




IDE68k kartı alttan görünüş, Xilinx CPLD

**Şekil 1.**

Projenin belkemiği Xilinx marka bir CPLD. Kaynak kodları proje sitesinden indirebilirsiniz. Proje kodları maalesef ABEL dilinde yazılmış. Bu kart ne yazıkki 'tak ve çalıştır' değil. Normalde Amiga 500'de kullanılmayan Gayle registerlerina erişildiğinde / OVR ve /INT2 sinyallerini kullanarak veriyolunu devralması gerekiyor. Bu iki sinyali Amiga 500'ün yan bölümündeki genişleme yuvasından çekebilirsiniz. Bu işlem ile ilgili daha ayrıntılı bilgiyi proje sayfasından alabilirsiniz.



IDE68k Amiga 500 içinde

**Şekil 2.**

Benim ilk izlenimlerim kartın büyük uyumluluk gösterdiği yönünde. Tam olarak bir A600/1200 IDE portu gibi çalışıyor, herhangi bir sorun ile karşılaşmadım. IDEFIX programı ile IDE portu üzerindeki CD-Rom'u tanıttım ve veri alışverişi yaptım. Bu aleti Action Replay ile kullanmayı düşünenleriniz için kötü bir haberim var. Action Replay (en azından bendeki MK3) Kickstart 37.300 ve üzeri ile uyumsuz. Dolayısı ile bu pek mümkün görünmüyor.

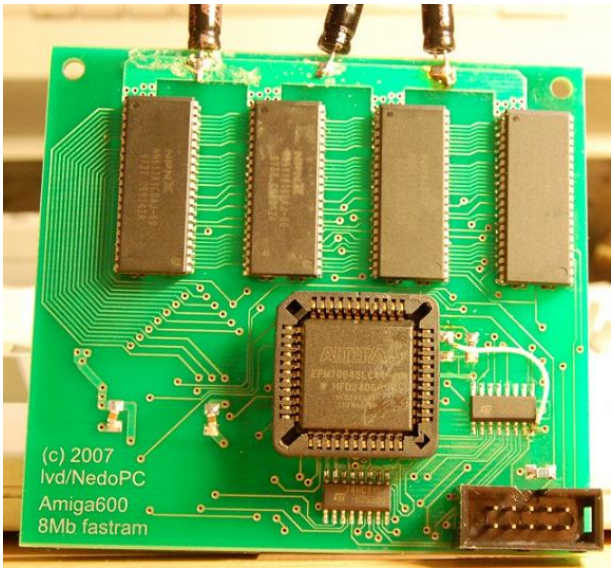
## Amiga 600 için 8 mb Fast RAM

Diğer bir projemiz ise Rusya'dan. Bu proje, Amiga 600'ün 68000 işlemcisi üzerine yerleştirilen bir kart ile sisteme 8 mb FastRam eklenmesi esasına dayanıyor. Amiga 600'ler piyasaya sürüldüklerinden beri genişlemesi en zor Amiga modellerindendir. PCMCIA yuvası ile sisteme 4mb'a kadar RAM eklemek mümkün, ama bu yöntem hem yavaş hem PCMCIA portu kapattığından pek tercih edilmiyor. Dolayısıyla bu proje Amiga 600 fanları için çok büyük önem taşıyor. Bu projede Altera marka bir CPLD Ram'ı kontrol etme ve sisteme tanıtma görevini üstlenmiş. Ram olarak ise 8 mb EDO ram'lerden sökülen 4 adet 16 bit 1Mword ram çipleri kullanılmış.

Amiga 600 Fast ram projesi, Amiga 600 üzerinde

**Şekil 3.**

Bu kartın resimlerde gördüğünüz 2007 versiyonu 8mb'a fixlenmiş bir tasarım. 68000 işlemcili sistemlerde genişleme kartlarının yerleştiği Zorro2 adres aralığı 8mb ile sınırlı olduğundan, 8mb'ın tamamının bir Ram kartı ile kapatılması aynı bölgeyi kullanan PCMCIA sürücüsünün devre dışı kalması anlamına geliyor. Amiga 600 için ise PCMCIA, gerek ethernet, gerek Compact Flash kartları aracılığıyla olsun, PC ile veri alışverişi için vazgeçilmez bir kaynak oluşturduğundan bazı durumlarda eklenen Ram'in 4mb indirilebilmesi büyük avantaj sağlıyor. Bu yüzden kartın yeni 2008 versiyonunda 4mb'a düşüş için bir jumper kontrolü eklenmiş durumda.




Kart üzerinden yakın bir detay

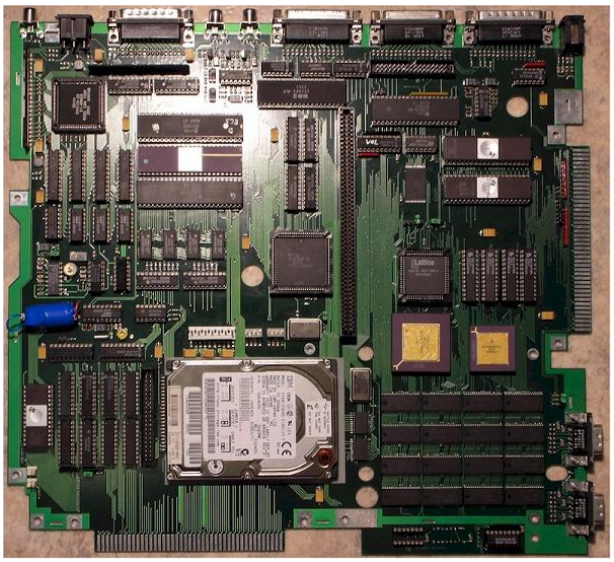
**Şekil 4.**

Bu projenin en önemli tarafı, 8mb FastRam için Amiga Autoconfig işleminin çözülmüş olması. Autoconfig, Amiga ile Amiga aparatları arasında tak ve çalıştır esasını sağlamak amacıyla geliştirilmiş bir protokol. Herhangi bir Amiga sistemine takılan genişlemeler, AutoConfig protokolü ile sistem ile haberleşip kendilerini sisteme tanıtabiliyor. Böylelikle Amiga kullanıcıları, zor dip switch/hafıza ayarlarından kurtulmuş oluyor. Bu protokolün 8mb ram için çözülmesiyle birlikte, diğer Amiga modelleri içinde Ram genişlemelerinin yolda olduğunu söyleyebiliriz. Özellikle bu proje ile ilk tanıttığımız IDE68k projesinin birleşmesi çok kullanışlı bir kartın ortaya çıkması anlamına geliyor.

# Amiga 1000 için Yeni Nesil Anakart Projesi

Sıradaki proje ise Almanya'dan Georg Braun tarafından geliştirilmiş. Bu proje diğer iki projeden farklı olarak satılması/çoğaltılması planlanan bir proje değil. Bu projede Amiga 1000 anakartı bir takım ek özelliklerle yeniden tasarlanmış ve imal edilmiş. Bu özellikler ise gerçekten dudak uçuklatacak cinsten. Sıradan başlayalım :

1. 68030 50 MHz işlemci ve 68882 FPU
2. 8MB 32Bit-Fastram
3. Scandoubler/FlickerFixer (Amber)
4. IDE sürücüsü
5. 2MB Chip Ram
6. Zorro 2 Yuvası



**Şekil 5.**

Bu özellikleri tek tek inceleyelim. 68030 işlemci Fast Ram'e 32 bit bir veriyolu üzerinden erişim sağlıyor. Kullanılan RAM Statik RAM olduğundan Fast Ram erişimi oldukça hızlı. Scandoubler/ FlickerFixer özelliği, Amiga 3000 ve A2320 kartlarında kullanılan Amber çipine dayanıyor. Bir şekilde bu çipten elde edip anakarta yerleştirmeniz gerekiyor. IDE sürücüsü, BSC'nin ürettiği AT-Bus 2008 tipi IDE sürücüsüne dayanıyor. Anakarta bu kartın ROM'unu takmanız gerekiyor. Manual Almanca olduğu için Kickstart ROM konusunu maalesef tam anlayamadım. Normalde iki adet 8 bit ROM'dan oluşan Amiga 1000 kickstartı iki adet 16 bit ROM soketiyle değiştirilmiş. Anladığım kadarıyla bunlardan biri orijinal (27c400 uyumlu) Amiga500/2000 Kickstart ROM'larını alacak şekilde, diğeri ise pinout'u biraz değişik olan 27C4096 ROM'ları alacak şekilde tasarlanmış. 27C4096 piyasada daha rahat bulunabilen bir ROM olduğu için bu seçenek eklenmiş olabilir.

Bu tasarımı aslında Amiga 1000 olarak adlandırmak biraz yanlış olabilir. Neredeyse bir Amiga 3000 kapasitesinde bir tasarım olmuş, Amiga 3000'den tek farkı, sistem veri yolunun 16 bit olması (Zorro2 kullanımından ve tek 16 bit Kickstart Rom kullanımından bunu çıkarımda bulunabiliriz). Bu projede dikkat ederseniz orijinal çipset olduğu gibi kullanılmış, yeni bir çip tasarımı yapılmamış. Anakart dizaynı ve tabi bunca ek birimin birbiri ile etkileşimini çözen bir proje. FPGA ve CPLD cinsi programlanabilir çipler kullanılmamış, GAL cinsi daha eski tip diyebileceğimiz programlanabilir çipler kullanılmış. Orijinal Amiga modellerinde de GAL tipi çiplerden görebilirsiniz. Bu GAL çiplerin program dosyaları da proje ile birlikte download edilebiliyor.

Projenin sahibi Georg Braun'un sayfasında bunun dışında birçok ilginç proje mevcut. Bu konulara ilgisi olan arkadaşların incelemesini tavsiye ederim.

Evet bu sayı için şimdilik bu kadar, gelecek Plazma'da yeni Amiga donanımlarıyla tekrar görüşmek üzere !

coze.00 (at) gmail (dot) com

* Amiga terminolojisinde Amiga 600, 1200 ve 4000'in IDE birimleri geçmiş programlar ile uyumluluk açısından scsi.device olarak adlandırılıyor.

Linkler:

IDE68k Projesi:

http://www.students.tut.fi/~leinone3/ide/ide68k.html

Amiga 600 için 8mb Fast Ram projesi:

http://nedopc.com/lvd/Projects/a600_8mb/index.html

Amiga 1000 yeni nesil anakart projesi:

http://www.gb97816.homepage.t-online.de/a1kboard.htm

Amiga 1000 yeni nesil projesinin Almanca forum topiği:

http://www.a1k.org/forum/showthread.php?t=8947

# Lord Henry'nin Köşesi

Özgür (Lord H. Wotton II) Yıldırım

## JASON BUSBY, NAMI DİĞER 3dBuzz, İÇİMİZDEN BİRİ.

3D ile ilgilenen arkadaşlarımız çok iyi bilirler, 3D' de "ücretsiz video tutorial" lar artık olmazsa olmaz bir hal aldı. İnsanlar takıldıkları bir noktanın hemen video tutoriallarını aramak yoluna gidiyor. Bir yığın "Professional training DVD" şöyle dursun, "screen capture" programlarının yaygınlaşması ve bedava hosting hizmeti veren web sitelerinin çoğalması ile bu ücretsiz video tutorial' lar tam bir fenomen haline geldi. Tabii bu tutorial' ların bir de anlatıcıları var. Kimilerinin ismi cismi belli değil, önemli de değil, kimileri ise gerek anlatım tekniği ve yeteneği, gerekse de ilgili uygulamaya olan hakimiyeti ile tam bir "star".

Bu starların en başında Alex Alvarez geliyor şüphesiz. Hollywood' daki pahalı bürosunda Silicon Graphics bilgisayarının karşısına geçip bön bön bakarak Istanbul' un varoşları dahil olmak üzere tüm dünyaya "üç boyutun sırlarını" anlatmış olan bu adam şöhretini sonuna kadar hak ediyor. Fakat size anlatacağım kişi Alvarez değil. Nedeni belki de Türk insanın sıcak kanlı ve "bizden biri" insanlara daha bir yakınlık duyması. O donuk ve bıkkın bakışları ve taa ABD' den monitörüme taşan negatif enerjisiyle (!) Alex maalesef bu şansı kaybediyor. Skate' in X2006' da insanların yüzlerinde sıcaklık araması gibi, izlediğim videolarda anlatıcıdan bir hoşluk, insani bir taraf bekliyorum.

Jason Busby. Asparagaz.com' un dediği gibi: O içimizden biri.

**Şekil 1.**

İşte tam bu noktada 15 yıllık programcılık geçmişinden sonra 3d alanına yoğunlaşmış bir "guru" yu, Jason Busby' yi tanıtmak istiyorum. Beyaz anglo sakson protestan görünümünün altında tam bir "bizden biri" var.

Bu adam video tutorial' ları yanına bir yorumcu alarak hazırlıyor. Aynı futbol spikeri ve yanındaki yorumcusu gibi, Jason Busby yanına bazen "Zakk" ya da başka birilerini alarak programı anlatıyor. Yorumlar yapılıyor, hikayeler anlatılıyor. Ve bu da video' yu inanılmaz keyifli hale getiriyor.

Jason Busby' nin eşi ve kızlarınında dahil olduğu 3dBuzz ekibi. Fotografta dünyanın dört bir yanına gönderecekleri yeni training dvd'lerini hazırlarken görülüyorlar.

**Şekil 2.**

Jason' un en önemli özelliklerinden biri de birçok 3d programıyla ilgili derin bir bilgi birikiminin olması. Hollywood' daki silicon graphics "background"' lu dinozorların tekelinde olan Houdini' ye bile kapsamlı bir video tutorial hazırlayarak bu alanda büyük bir boşluğu doldurdu. Motion builder serisiyle de gerek kendisi gerekse yardımcısı "Zakk" benden tam not aldı. Motion builder' i neredeyse (!) en ince ayrıntısına kadar anlattılar.

Üstelik Jason aslen bir programcı ve oyun programcılığı alanında deneyimli.

Jason ve ekibinin ortamını aşağıdaki youtube video'sundan daha iyi hiç bir şey anlatamaz. Izleyin.

http://www.youtube.com/watch?v=GreqRHOt-nI

Unutmadan,

www.3dbuzz.com

3D'ye başlayacaklar için, çok ama çok önemli bir kaynak...

# DEBRIS TARTIŞMALARI

Debris ile ilgili tartışmalar sürüp gidiyor. Beğenmeyen arkadaşlar bir de Farbrausch' un minus-03: Farbomat' ı denesinler. (!)

# ZAMAN TÜNELİ

Umut "Detay/Clique" Çelenli' den geliyor:

(....) içinde görmeye alıştığımız şeylerden hiç yok. Ne plazmalar ne sinüs plotler, ne sıkıcı vektör küpler. Hiçbiri yok! Bir scroller bile yok içinde. Demoda sadece insanlar dans ediyor o kadar. Fakat Öylesine güzel hazırlanmış ki, ben şimdi ne diyeyim? (...) Coder zümresi bu demo sayesinde kolaylıkla kafayı yiyecektir. (Bizim gibi.) -Amiga Dergisi Sayı 4 S: 32' de Spaceballs State of the Art' ı tanıtırken-

(....) Bu demolarında da yine insanlar (daha çok kızlar) dansediyor, ve işin güzeli BU SEFER ONLARI GÖRÜYORSUNUZ. Kızların gülümsediğini görüyorsunuz!!!!! (....) Bence demo piyasası bu gibi gruplar sayesinde daha çok insana hitap edebiliyor. Ben şahsen bu gibi programlamayı nokta basma rekorunu kıran bir demoya tercih ederim... -Amiga Dergisi Sayı 11 S:14' de Spaceballs' ın 9Fingers' ını tanıtırken-

# GOAL!

**Şekil 3.**

Eskiden Goal! diye bir oyun vardı. Bunu diyorum çünkü görünüşe bakılırsa herkes kickoff 2 diyor başka bir şey demiyor. Goal! çıktığında hemen gidip almıştık. (Çektirmiş miydik yoksa yahu? :D) O günden sonra kickoff 2 disketleri rafa kalkmıştı. Arkadaşlar arasında yeni trendimiz Goal! dü. Yıllarca bunu oynadık. Internetli yıllardan (!) sonra bir baktım Goal!' ü kimse "takmıyor." Nedenini anlamak gerçekten zor!!

**Şekil 4.**

Dinodini bir zamanların unutulmaz futbol oyunu programcısı idi. Şimdilerde bir jazz grubunda müzisyenlik yapıyor.

Evet ben de goal' de orta sahadan gol atabiliyordum tabii ki... Hem de doksana. O çatala girince direklerden tiiiink diye çok tatlı bir ses çıkardı. Kick off 2 den çok daha gelişmiş ve zevkli bir oyundur bu yahu.

Şimdilik bu kadar arkadaşlar. Kalın sağlıcakla.

Lord Henry Wotton II

# Drey'in Sequencer Köşesi

Emir (Drey) Diril

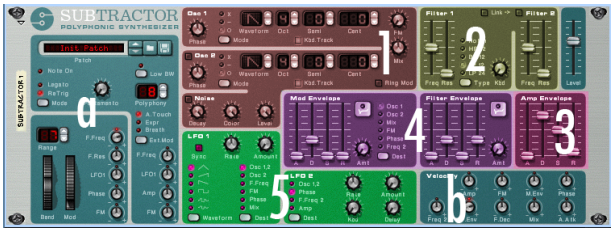
Tüm Plazma okurlarına merhaba! Sequencer' lara adanmış olan bu köşede tanıtımlarımıza kaldığımız yerden devam ediyoruz. Yazı dizimize ilk olarak Reason 3.0 ile başlamış ve programa derdimizi nasıl anlatacağımıza dair genel bir giriş yapmıştık. Artik Reason'a biraz daha ısınmış olduğumuzu düşenerek detaylara geçebiliriz. İncelemeye ilk olarak Reason' ın SUBTRACTOR adlı synth' inden başlamak istiyorum.

## SUBTRACTOR

Subtractor ve Reason penceresinin genel görünümü

**Şekil 1.**

Öncelikle Subtractor' ın tam olarak ne olduğunu tanımlayalım. Subtractor klasik bir "analog, subtractive" synth modellemesidir ve mono çıkışlıdır. "subtractive" demek o synth' de ses preset' lerinin, üretilen sesin çeşitli filtrelerle yontulması sonucu elde edildiğini gösterir. Bu yontulmanın da ne anlama geldiğini birazdan Subtractor' ın bölümlerini incelerken göreceğiz. İncelememizi daha kolay bir hale getirmek için önce aşağıdaki şemayı inceleyelim ve görelim bakalım Subtractor bize neler sunuyor ;)



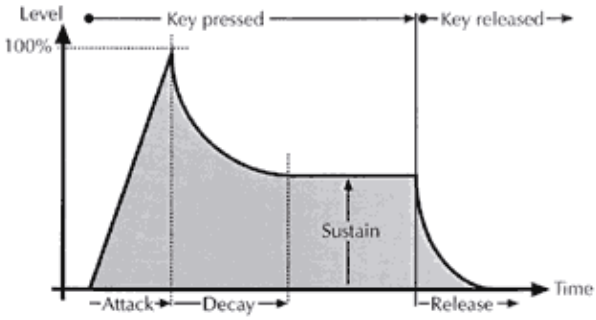

Subtractor' daki ana bölümler

**Şekil 2.**

"1" numaralı bölümden incelemeye başlayacak olursak buranın Subtractor' ın kalbi olduğunu söyleyebiliriz. Burada 3 adet oscillator yani ses üreteci bulunur. Bunların ilk ikisi bize temel dalga formlarından birini seçme imkanı verir. Yani chip müziği ile uğraşanların da hemen tanıyacağı "sinus, üçgen, kare, testere" nin de yani sıra yaklaşık 30 tane daha temel ses dalgasından birini seçebiliriz. Bunun dışında sesimizin temel karakterlerinden bazılarını da yine burda seçeriz. Phase knob' u ile seçtiğimiz dalganın faz aralığını ayarlayabilir, "oct, semi, cent" bölümleriyle dalga formumuzun ince akordunu yapabiliriz. Üçüncü ses üretecimiz de Noise diye tabir edilen ve yine chip müziğinde yaygın olarak kullanılan bir ses tipini üretir. Yine bu sesin karakterini de hemen yanındaki üç knob ile değiştirebiliriz. Son olarak "1" numaralı bölümdeki tüm bu seslerin hangisini kullanacağımızı sol üst köşelerindeki kırmızı kare buton aracılığıyla belirleyebiliriz ( OSC 1 daima açıktır) ve seçili olan üreteçlerin karışım miktarlarını Mix butonu ile belirleyebiliriz.

"2" numaralı bölüm, üretilen basit dalga formumuzu ufak ufak şekillendirmeye başladığımız kısımdır. Burada iki tane filtremiz vardır. Filter 1 her zaman devrededir, Filter 2 ise isteğe bağlı olarak sol üstündeki kırmızı kare buton ile devreye dahil olmaktadır. Burada önümüze beş adet filtre tipi sunulmuştur. Bunların arasından seçeceğimiz tipe göre hemen solundaki iki slider ile filtrenin çözünürlüğünü (res) ve frekans toleransını (Freq) belirleriz. Ayni şeyler ikinci filtre için de geçerlidir ( devre içi olduğu müddetçe). Burada yaptığımız işlem, seçilen filtre özellikleri dahilinde, dalga formundaki fazlalıkların traşlanmasıdır. Filtreden geçirilmiş bu yeni ses dalgası bir sonraki kısıma, yani yükselteç zarfına (amplify envelope) aktarılır.

"3" numaralı kısım sesin dinamik şekli (yani nasıl başlayıp süreceği ve nasıl sonlanacağı) açısından en önemli kısımdır. Yaratmış olduğumuz dalga formunun tam olarak nasıl bir amaca hizmet edeceği (bir lead sesi ya da bir pad) buradaki ayarlara bağlıdır. Tabi bunun için öncelikle synth'ler için önemli bir bölüm olan "envelope" yani "zarf" kavramını anlamak gerekir. Bunun için hemen envelope şemasına bir bakalım.




Synthisezer için Zarf (envelope) diagramı

**Şekil 3.**

Bir synth zarfı dört ana bölümden oluşur. "attack" bölümü sesin sıfırdan maksimum değerine ne kadar zamanda ulaşılacağını gösteren bölümdür (hatta bazı synthlerde bu değerin lineer ya da logaritmik olarak artması bile kontrol edilebilir). Burada ki süreyi uzatarak sesi daha mistik bir hale ya da kısaltarak daha agresif bir hale getirebilirsiniz. "decay" bölümü, ses maksimum notkasına ulaştıktan sonra "sustain" denilen kısma geçene kadar nasıl ve ne kadar düşeceğini gösterir. "sustain" de ise sesin uzamaya devam ettiği kısımdır. Yani burayi tuşa bastıktan sonra parmağımızı çekene kadar geçen uzama olarak düşünebiliriz. Parmağımızı çektikten sonra da sesin nasıl sonlanacağı "release" bölümünde belli olur. Yine burada da release kısmını uzatarak sese hafif bir reverb efekti katılmış izlenimi verebiliriz.

Bu noktada, zarfımızdaki bu dört ana değerle oynayarak filtrelenmiş sesimizin gövdesini isteğimize göre şekillendirmiş oluyoruz, ve bir sonraki bölüm olan "mod" ve "filtre" zarflarına geçiyoruz.

Geldik bölüm “4” e. Bu kısımdaki birinci zarfımız “filtre zarfı”. Yani bölüm “2” deki filtrelerimizin nasıl bir şekilde devreye gireceğini gösteren zarf. Yine filtre zarfında “attack” kısmını kısaltarak filtrelerimizin devreye girmesini geciktirebilir ve parlak olarak başlayan sesin sonradan matlaşmasını sağlayabilirz. İkinci zarfımız ize “mod zarfı”. Buradaki mod kelimesi modulationın kısaltılması olup standart bir midi keyboard'daki iki kontrol tekerleğinden birini ifade eder ( birincisi modulation wheel, ikincisi de pitch wheel’dir). Bu zarf ile de (diğer zarflarda da olduğu gibi) seçili modülasyonun devreye nasıl dahil olacağı belirlenir. “2” numaralı bölümdeki iki önemli kontrol çeşidi de zarfların sağ alt köşelerindeki “amt” knobları ve sağ üst köşelerindeki gri kare butonlardır. “amt” knobları ile zarflarımızın ne oranda etkin olacaklarını (amount) seçeriz. Gri kare butonlara gelince. Bunlar biraz enteresan bir iş için. Bu düğme aktif hale gelince ADSR (attack,decay,sustain,release) ayarlarını tepe taklak edip tam tersine çevirir. Presetleri yaratırken çeşitliliği arttırmak adına bir kolaylık ;)

Sesimize şekil verme ile ilgili son bölüm, “5” numaralı bölüm. Burada ik adet LFO’muz var yani “Low-frequency oscillator”. Peki bununla ne yapabiliriz. Bir kere bu kısımda yine bize basit dalga formları üreten iki oscillator’ümüz var. Ama bunların bölüm “1” dekilerden farkı ürettikleri dalga formlarının frekansı düşük olduğu için seslerinin duyulmaması. Peki bu sessiz dalgalar ne yapıyor? Bu sessiz dalgalar yine aynı bölümde bulunan dalga tipi (waveform) ve hızı (rate) ayarları ile düzenlendikten sonra “dest” yazan kısımdan (destination) seçilen bir hedefe yönlendiriliyor. Yönlendirildikleri hedefin karakteristiğini kendi dalga şekilleri doğrultusunda etkiliyorlar. Ne miktarda etkileyecekleri de yine bölüm “5”deki “amt” knobları ile belirleniyor. “sync” butonu ile de dalga formunun hızı şarkının ritmine eş zamanlı hale getiriliyor. Örnekle açıklamak gerekirse triangle waveformu seçip amountu sonuna getirir ve dest’ I de osc 2 seçersek, 2 numaralı ses üretecinden üretilen sesin lineer olarak 2 oktav aralığında değiştiğini duyarız. Ayrıca bir de sync butonunu aktif hale getirdiğimizi ve rate knob’unu da ¼ değerine getirdiğimizi düşünelim. Bu durumda bu lineer salınımın her bir adımı şarkının bpm’ine uygun 4’lük nota zamanında gerçekleşecektir.

Son olarak “a” ve “b” bölümlerinden bahsetmek istiyorum. Bu bölümlere numara yerine harf vermemin sebebi, bunların sesin üretiminden daha ziyade performansa yanı çalım esnasında olacaklara yönelik ayarlar olması. “a” bölümündeki en önemli ayarlardan biri “polyphony”dir. Yani aynı anda kaç farklı tuştan ses alınabileceği burdan belirlenir. “portamento” knob’u ile bir notadan diğerine geçerken aradaki geçiş süresinin hızı belirlenir. Bu knob saat yönünde ilerledikçe bu zaman artar. Örnek olarak portamentoyu “8-10” gibi bir aralığa getirip polyhpony’yi de 2’ye düşürürseniz, Daha kaygan ve monophonic bir solo sesi üretebilirsiniz. “bend” tekerleğinin üstündeki “range” kısmından, tekerleği sonuna kadar çevirince sesin en fazla ne kadar bükületeceği belirlenir ( burada 7 yazıyorsa ses toplam olarak 7 yarım ses kadar daha aşağı ya da yukarı bükülecektir). Mod tekerleğinin sağ yanındaki ayarlar ise tekerlek hareket ettiğinde hangi parametrenin ne miktarda değişeceğini seçmek içindir. Son olarak “b” bölümü, yani “velocity” (şiddet) bölümü. Çalım sırasında tuşlara basma şiddetimize bağlı olarak hangi parametrelerin değişmesi gerektiğini ayarladığımız bölümdür. Burdaki yapacağımız ayarlara göre örnek olarak filtrelerimizin daha yavaş basınca daha az etkin daha sert basınca da daha çok etkin olmasını sağlayabiliriz.

Tüm bu laf kalabalığından sonra eminim somut örnekler daha da pekiştrici olacaktır. Bu yüzden, ekte, sizler için hazırladığım Reason dosyasına göz atmak isteyebilirsiniz. Orada, tarafımdan hazırlanmış presetler, sizlere Subtractor’ ın değişik kullanım alanları ile ilgili fikir verecektir diye düşünüyorum. Ayrıca, her ne kadar modern bir platform olsa da, Reason’da Subtractor ile de “oldskool” tınılar yakalanabildiğini görebilirsiniz ;)

Bir sonraki sayıda görüşmek üzere, keyifli müzikler :)

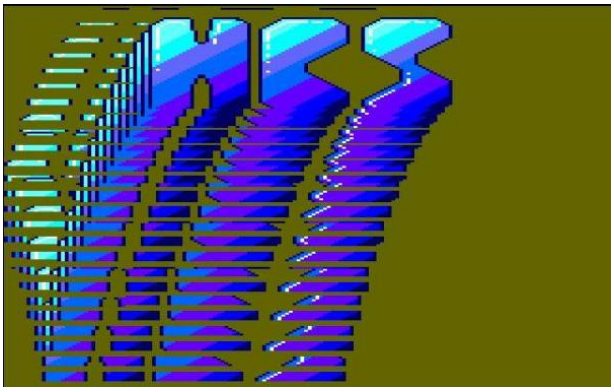
Emir “Drey” Diril

# Amstrad CPC Demoscene

Türker (Alcofribas) Gürevin

Geçen sayıda CPC dünyasına kısa bir giriş yapıp, hem yerel hem de global olarak CPC kullanıcılarının hangi süreçleri takip ettiğini ve CPC uygulamalarını hızlıca deneyebilmek için ihtiyacımız olan temel program-komut türlerinden bazılarını tanıtmıştık. Meraklıları zaten çoktan yeni ufuklara doğru yelken açmıştır; kimi oyunlara göz atmıştır, kimi de acaba günümüzde CPC için neler yapılıyor sorusuna yanıt aramaya başlamıştır. Biz ise bu sayı ile beraber, birkaç sayılık bir yazı dizisi ve sonrasında güncel çalışmaları tanıyacağımız bir sürece başlıyoruz. Konumuz, bir bilgisayar platformundaki en keyifli faaliyetlerden biri olan "demoscene" kavramı ve buna bağlı olarak CPC ortamında üretilen demoların tarihsel gelişimi. Elbette ki konuya yabancı olanlar için "demo" veya "demoscene" kavramı pek bir şey ifade etmeyebilir. Bu kavramların anlamı ve içeriğiyle ilgili olarak Plazma'nın önceki sayılarına bakmanızı tavsiye ederim, birbirinden güzel ve açıklayıcı yazılar bulacaksınız.

Diğer yaygın platformlar olan C64 ve Sinclair Scene'i kadar olmasa da CPC dünyasında da dikkate değer oranda demo üretimi olmuştur ve hala olmaktadır. Başlangıç noktası olarak kabul edebileceğimiz ilk CPC demosu ise Alman MCS grubuna ait olan "MCS Demo5" tir. Tabii 5 rakamı hemen dikkatinizi çekmiştir. Neden 5 sorusunun yanıtı ise grubun Demo5'den önceki çalışmalarının pek de demo olarak sınıflandırılabilecek nitelikte olmamasıdır. Bunlar daha çok bir "intro" havasında idi. Bahsettiğimiz bu 5 nolu demonun içeriği ise şunlardan ibarettir; renk geçişlerinin olduğu bir fon üzerinde yukarı aşağı hareket eden ve dönen şeffaf küreler, sıçrayıp duran devasa bir MCS logosu, raster ve kayan yazılar. 1988 yılına ait olan bu çalışma o dönem için kabul edilebilir bir içerik ve kaliteye sahip.



MCS Demo5

**Şekil 1.**

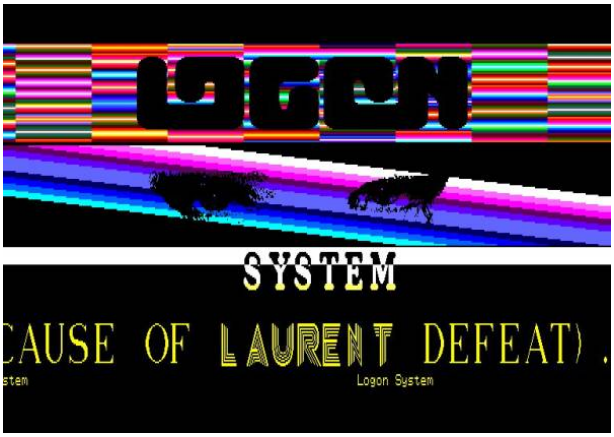
Almanların bu güzel çalışmaları, Fransızların da uyanmasını ve önlerinden henüz geçmiş ve kaçmakta olan bu treni yakalamaları için tetikleyici olmuştur. Bu bağlamda, Fransızların ilk saygın çalışmaları Fred Crazy ve Dr. TKC tarafından yapılmıştır.


Dr.Tkc Demo 2

**Şekil 2.**

Bu iki saygın ismin demoları, küçük olmalarına rağmen birçok önemli tekniğin ilk defa kullanıldığı ve CPC'lerin yeteneklerini ortaya koyan çalışmalar olmuştur. Özellikle Dr. TKC'nin çalışmalarında, muhteşem "Yatay Donanımsal Scroller" kullanılmış ve hız konusunda sınırlar (her kaydırmada 2 byte) zorlanmıştır. Ne yazık ki Dr. TKC kısa bir süre sonra CPC demoscene'den ayrılmıştır.

Ve bir diğer "donanımsal kaydırma" tutkunu olan Longshot da yine bu sıralarda, CPC demoscene'de yerini almak üzere ürünlerini sunmaya başladı. Bu tarihlerde sadece bir üyesi olduğu için tam olarak "Logon System" den bahsetmek mümkün değil. "The Longshot Demo" isimli çalışması, görkemli bir "donanımsal kaydırma" ve "PC'den aktarılmış yazıtipleri" ile diğer çalışmaları için bir öncül olmuştur. Bu demoda David Whittaker'ın Glider Rider oyunundaki müzikleri kullanılmıştır. Kendisi için bir başyapıt ve zamanına göre devrim olarak sınıflandırabileceğimiz "Revolog" isimli çalışmasında ise; yatay-dikey donanımsal kaydırma, PC yazıtipi, yatay rasterlar, split rasterlar ve dahası Atari ST'den birebir kayıpsız olarak aktarılmış müzikler kullanılmıştır. Tabii müziklerin Atari ST'den aktarılması olayından bahsedince bunu ilk defa yapanın Longshot olmadığını belirtmekte de yarar var. Ayrıca, başarılı 2 demosundan ve Titus'dan önce, çalışmalarının birinde overscan ekran da kullanmıştır. Şunu bilmek gerçekten hoş ki; gariban bir CPC üzerinde overscan programlamak 16 bit olmasına rağmen Atari ST üzerinde programlamaktan çok daha kolaydır.




The Longshot Demo

**Şekil 3.**


Malibu Demo 4

**Şekil 5.**


Revolog

**Şekil 4.**

Longshot ile yine aynı dönemlerde(1989), Malibu Crackers'ın çalışmaları da yavaş yavaş CPC demoscenede kendine yer edinmeye başlamıştı. Bu grubun gerçek değeri, özellikle "Malibu Demo 4" ile çok daha iyi anlaşıldı. Overscan bir ekran boyunca kaydırma, diferansiyel kaydırma ve çoklu ekran modu kullanımı bu demonun en temel teknik özelliklerini oluşturmaktadır. Bu grubun demirbaş programcısı olan Naminu, bir süre sonra Logon System grubuna katılmıştır.

Fefesse bizi hayretlere düşüren "Yao Demo" isimli çalışmasından önceki ürünlerinde programcılık seviyesi olarak biraz zayıf kabul edilmektedir. Bu çalışmalarında "dalga kaydırmaları" güzel kullanmışsa da müziğin devreye girmesi ile beraber demolarda bir yavaşlama ve sıkıntı göze çarpmaktadır. Gelgelelim "Yao Demo" ise; derinlik duygusu yaratan küreler, yatay-dikey ve dama tahtası şeklinde donanımsal kaydırmaları ile gerçekten 90 öncesi döneme ait iyi bir çalışmadır. Teknik özelliklerinin yanı sıra bu demo ile beraber Fefesse ve Longshot arasındaki savaş da su yüzüne çıkmıştır, ki daha sonra Alman ve Fransız demo programcıları arasında ortaya çıkacak olan kadar kanlı bir savaştır bu.


Yao Demo

**Şekil 6.**

Demolarında yeni bir teknik getirmeseler de, P007'nin ancak 128K'lık bir CPC ile yapabildiği "mad sinus scroller" ı 64K'lık bir CPC üzerinde başarılı ile gerçekleştirerek kendilerine haklı bir

ün edinen GPA grubunu da unutmamak gerekli. Yine aynı dönemde faaliyet gösteren Krad'os Cracker ve ünlü "Trash Demo" ürünleri; muhteşem grafikleri(Zeigboss), çılgın topları ve Mister Plus'ın iyi programcılığı (mode 0 da overscan ve mode 2 bir sürü raster) ile de saygıyı hak etmektedir.


Trash Demo

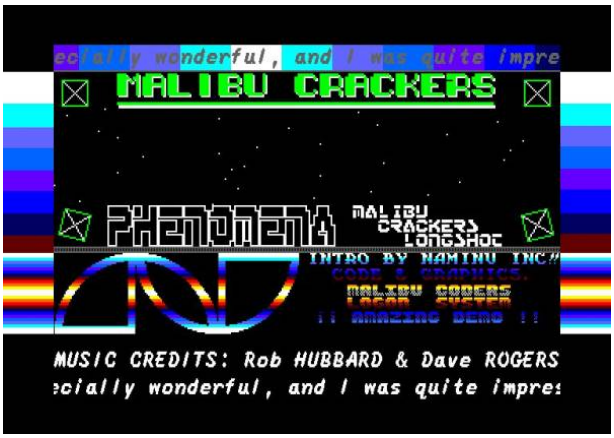
**Şekil 7.**

Şu ana kadar bahsettiğimiz devirde, Almanya ve Fransa CPC demoscene alanında iki temel olmuştur. Ancak, N.W.C(New Way Cracking) grubunun girişi ile beraber Danimarka da bu konularda söz sahibi olduğunu ortaya koymuştur. Her biri birbirinden güzel rasterlar ve gerçekten övgüye layık müzikleri ile pek çok demoseverin gönlünde taht kurmuşlardır. Bir diğer Danimarka kökenli grup olan JLCS ile aralarında sıkı bir rekabet de oluşmuş ve bu ürünlere de yansımıştır. N.W.C'nin "Kill JLCS" isimli demosunun mesajlarında bu açıkça görülmektedir. Ayrıca yine bu çalışmada, çizilmiş büyükçe bir logonun havada bir bayrak gibi dalgalanması çok başarılı bir şekilde uygulanmıştır.

Kill JLCS

**Şekil 8.**

Tekrar Fransa'ya dönecek olursak, daha önceki satırlarda bahsettiğimiz gibi Naminu'nun Logon System'e katılımı ve 3 aylık bir çalışmanın ürünü olan tam bir disketlik(sadece bir yüzü) "The Amazing Demo" isimli ürünleri ile yine bir devrim niteliğinde çalışma ortaya çıkmıştır. İçerik olarak; her yöne doğru onlarca kaydırma, ST'den aktarılmış ama örneklenmiş sesler kullanan müzik hemen akla gelen bazı güzel tekniklerdir.




The Amazing Demo

**Şekil 9.**

Bir diğer örneklenmiş seslerin kullanılması ürünü olarak, CCC grubundan CJC isimli programcının Amiga örneklenmiş seslerinden oluşan çalışması "Mega Sound Amiga"sını söyleyebiliriz.

Şimdilik burada duralım ve biraz soluklanalım. Sizin de gerek resimlerden gerekse demoların bizzat kendilerinden fark edeceğiniz gibi büyük bir değişimin ya da diğer bir deyişle ilerlemenin eşiğindeyiz. Tarih olarak 80'lerin sonuna ve 90'ların hemen başına gelmiş bulunuyoruz. Bir sonraki yazımızda devam etmek üzere…

Kaynakça:

http://www.cpcscene.com

http://cpcrulez.free.fr

http://www.pouet.net

Bana ulaşmak isterseniz:

8bitmicro (at) gmail (nokta) com

# Atari 800 XL/XE - Bölüm 1

Emir (Skate) Akaydın

## Giriş

80'li yılların ortalarında, ufacık bilinçsiz bir çocukken tanıştığım Atari 800 XL ile yaşadığım çocukluk aşkının 20 seneden uzun bir mazisi vardır. Ancak ne yazık ki aramızdaki elektrik kalıcı olamadı. Atari, bu 20 seneyi aşkın sürenin en fazla 2-3 senesinde benim hayatımda aktif olarak yer aldı. Çocukluk yıllarımda daha ilkokula gitmezken tanıştığım bu cici bilgisayar beklenenden erken öksürüp tıksırmaya başlayınca, o yılların en popüler bilgisayarı Commodore 64 hayatımda girdi. Artık Atari'nin Commodore ile mücadele etmesi söz konusu değildi. Duygusal Atarim, Commodore ile olan aşkımıza olan kıskançlığından dolayı kısa süre sonra kör oldu (GTIA çipi yandı). Ben de Commodore ile olan aşkımdan kör olmuş olsam gerek, o güzelim bilgisayarı amelyat ettirmeye tenezzül etmedim ve diri diri gömdüm. Ancak yıllar sonra ruhu intikam için geri döndü (Atari800WinPlus Emulatörü).


**Şekil 1.**

Atari'yi terk etmemde geçerli sebeplerim olduğunu düşünmüştüm hep. Ne de olsa Atari çok daha eski ve ilkel bir bilgisayardı. Commodore 64 ise geleceğin teknolojisi idi. Commodore her konuda Atari'den üstündü. Atarinin o kötü grafikli oyunlarının Commodore versiyonlarıyla mücadele etmesine olanak yoktu. Ne de olsa Commodore hız, görüntü, ses konularında Atari'den kat kat üstündü. Kim takar Atari'yi!!!

Bir çocuğun bakış açısından böyle bir imaj oluşturduğu için Commodore firmasını binlerce kez tebrik etmek lazım. Çünkü yukarıdaki paragrafta geçen düşüncelerden çok azı gerçeği yansıtıyor. Evet, Atari 800 serisi Commodore 64'den yaklaşık 3 sene önce piyasaya sürülmüştü. Yani takvimsel olarak daha eski bir teknolojiydi. Evet, Commodore 64'ün SID çipi eşsizdi. Atari'nin POKEY çipinin sağladığı ses özellikleri ne olursa olsun, sonuçta kulağa ulaşan ses hep Commodore 64'de daha kaliteliydi. Evet, Commodore 64'de 25 satır yani 200 pixel yükseklik varken Atari'de 24 satır, 192 pixel yükseklik mevcuttu. Ancak bunların haricinde Atari, Commodore'dan birçok açıdan çok daha üstün bir bilgisayardı.

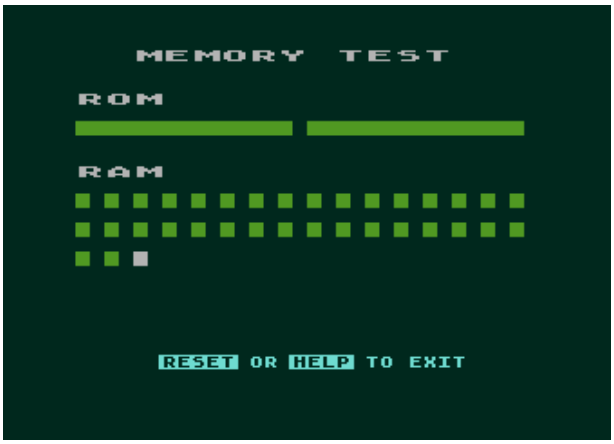
Şimdi sırayla Atari'nin sahip olduğu üretim tarihine göre çok üstün olan özelliklerini inceleyelim.


"Bırakın beni, ben kendi kendimi test ederim" diye haykıran, test programları içinde gelen eşsiz aletin ana test ekranı

**Şekil 2.**

## Atari 800 Serisinin Donanım Özellikleri

Atari, ana işlemci olarak 6502, görüntü için ANTIC ve GTIA, ses ve giriş/çıkış birimleri için ise POKEY çiplerini kullanır. Atari'nin işlemcisi 1.79 Mhz hıza sahiptir. Hafızası 64K RAM ve 24K ROM'dan oluşur. Cihazın üzerinde 2 joystick portu, çeşitli ek donanım ve kartuş portları bulunmaktadır.




**Şekil 3.**

Atari, 6502 mimarisini olduğu gibi almış ve kullanmıştır. Commodore 64'de olduğu gibi 6502'ye giriş/çıkış birimleri ekleyerek 6510 gibi özel bir işlemci geliştirme yoluna gidilmemiştir. Bunun yerine POKEY çipi hem ses, hem de seri giriş/çıkış işlemleri için kullanmak tercih edilmiştir.

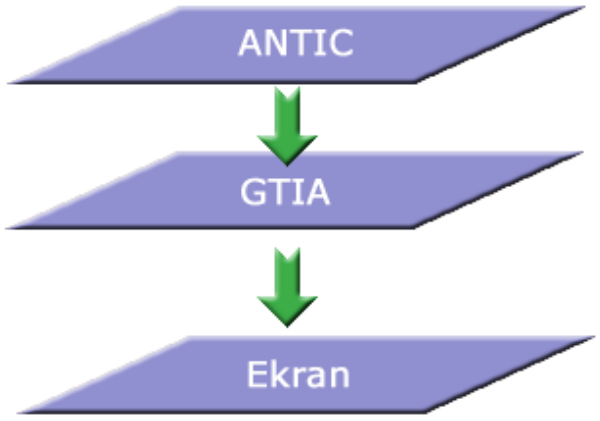
# Grafik Özellikleri

Atari grafik ve donanım destekli yaratıklar (spritelar) için iki ayrı çipten destek alır. Aslında yapı itibariyle grafiklerin iki katmandan oluştuğunu söyleyebiliriz.

ANTIC çipi Atari'nin bütün text ve grafik modlarını, yatay ve dikey ekran kayması gibi donanım destekleri sağlayan çiptir. Bu çipi en güçlü kılan özellik görüntü listesi (display list) adı verilen, Amiga'nın Copper'ına benzer bir özelliğe sahip olmasıdır. Bu söz konusu görüntü listesi ile her bir satırda (raster/tarama satırı) gösterilecek grafik/text modu değiştirilebilir. Commodore 64'de $d012 tarama adresiyle zahmetlice yapılan birçok iş, bu özellik yardımıyla zahmetsizce yapılabilir. Ayrıca bu özellik, Atari'nin basic dilinden de kolayca kullanılabilme avantajına sahiptir. İlerleyen bölümlerde bu konuyu tekrar ele alacağız.

ANTIC çipi elde ettiği görüntüyü GTIA (ilk üretilen Atarilerde CTIA) çipine aktarır.

GTIA (George's Television Interface Adapter) çipinin asli görevi görüntü sinyallerini televizyona iletmektir. Ancak bu sinyalleri televizyona ulaştırmadan önce görüntünün üzerine yaratıkları basan da yine bu çiptir. GTIA'in yetenekleri bununla da sınırlı kalmaz. Yaratık çarpışmaları, öncelik sıraları ve tüm grafik verilerinin (ANTIC çipinin sağladıkları da dahil olmak üzere) parlaklık kontrolleri bu çip tarafından kontrol edilir. Çipin yeni modellerinde sağladığı bir diğer özellik ise ANTIC çipine ek olarak 3 yeni grafik modunu kullanıma açmasıdır. Bu grafik modları daha fazla renk kullanımına izin verir ancak çözünürlükleri oldukça düşüktür (80x192).




**Şekil 4.**

Atari 800 XL toplam 16 farklı grafik modu destekler. Atari 800 XE modelleri ise * ile işaretlenmiş 4 adet grafik modunu desteklemezler. Desteklenen grafik modları aşağıdaki tabloda verilmiştir.

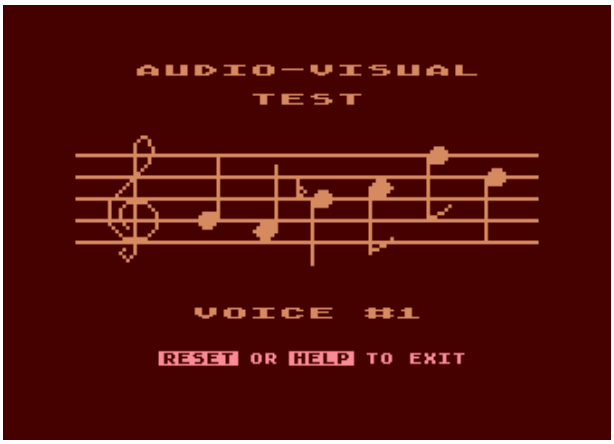
| Grafik Modu | Açıklama | Çözünürlük | Renk | Parlaklık | Tarama Satırı | Byte/Satır |
|---|---|---|---|---|---|---|
| 0 (Antic 2) | 1x1 text | 40x24 | 1 | 2 | 8 | 40 |
| 1 (Antic 6) | 2x1 text | 20x24 | 5 | - | 8 | 20 |
| 2 (Antic 7) | 2x2 text | 20x12 | 5 | - | 16 | 20 |
| 3 (Antic 8) | 4 renk grafik | 40x24 | 4 | - | 8 | 10 |
| 4 (Antic 9) | 2 renk grafik | 80x48 | 2 | - | 4 | 10 |
| 5 (Antic A) | 4 renk grafik | 80x48 | 4 | - | 4 | 20 |
| 6 (Antic B) | 2 renk grafik | 160x96 | 2 | - | 2 | 20 |
| 7 (Antic D) | 4 renk grafik | 160x96 | 4 | - | 2 | 40 |
| 8 (Antic F) | Tek renk yüksek çözünürlük | 320x192 | 1 | 2 | 1 | 40 |
| 9 (GTIA 1) | 16 parlaklık seviyesi | 80x192 | 1 | 16 | 1 | 40 |
| 10 (GTIA 2) | 9 renk grafik | 80x192 | 9 | - | 1 | 40 |
| 11 (GTIA 3) | 16 renk grafik | 80x192 | 16 | - | 1 | 40 |
| 12* (Antic 4) | Çok renkli 1x1 text | 40x24 | 4 | - | 8 | 40 |
| 13* (Antic 5) | Çok renkli 1x2 text | 40x12 | 4 | - | 16 | 40 |
| 14* | 2 renk | 160x19 | 2 | - | 2 | 20 |

| | | | | | | |
|---|---|---|---|---|---|---|
| (Antic C) | grafik | 2 | | | | |
| 15* (Antic E) | 4 renk grafik | 160x192 | 4 | - | 2 | 40 |

Tablo 1

* = Yalnızca Atari 800 XL'in desteklediği grafik modları

# Ses Özellikleri




**Şekil 5.**

POKEY çipi 4 adet 8 bit ses kanalı desteği sağlar. Ses kanallarının her biri kendi bağımsız frekans, dalga biçimi ve ses düzeyi parametrelerine sahiptir. Daha kaliteli ses elde etmek için 8 bitlik 4 ses kanalı yerine, kanalları 2'şer 2'şer gruplayarak 16 bitlik 2 ses kanalı elde etmek de mümkündür. Hatta istenildiği taktirde 1 adet 16 bitlik ve 2 adet 8 bitlik toplamda 3 ses kanalı kullanma olanağı da vardır.

# Giriş/Çıkış Desteği

POKEY çipi ses desteğinin yanı sıra tüm giriş/çıkış işlemlerinden de sorumlu olan çiptir. Özelilkle klavye ile ilgili 6 bitlik bir tarama desteği sağlar. Yani 64 tuşa kadar tuş kontrolü yapabilir. Bunların yanı sıra CTRL, SHIFT, BREAK gibi özel tuşları da farklı biçimlerde algılayabilir.

SKRES, SEROUT, SERIN, SKCTL, SKSTAT isminde 5 farklı register; BREAK, K, SIR, ODN, XD, T1, T2, T4 şekilnde 8 farklı IRQ kesmesi içerir.

Bunların yanı sıra POKEY'de ses kanallarını kullanan 3 adet sayaç mevcuttur. Bu sayaçlar kullanılmaya başlandığında ses kanalları resetlenir. Bu sayaçlar genellikle rastgele sayı üretmek için kullanılmaktadır.


**Şekil 6.**

# Ek Donanımlar

Atari bu konuda oldukça esnek tasarlanmıştır. 1088 KB'a (>1MB) kadar ek hafıza destekler. Standart kaset donanımın yanında 5.25 disk drive desteği de mevcuttur. Ayrıca aynı senelerde çıkan diğer birçok platformun aksine Atari'nin ek donanımları standartlaşmış ve bu donanımları destekleyen birçok uygulama ve oyun yayınlanmıştır.

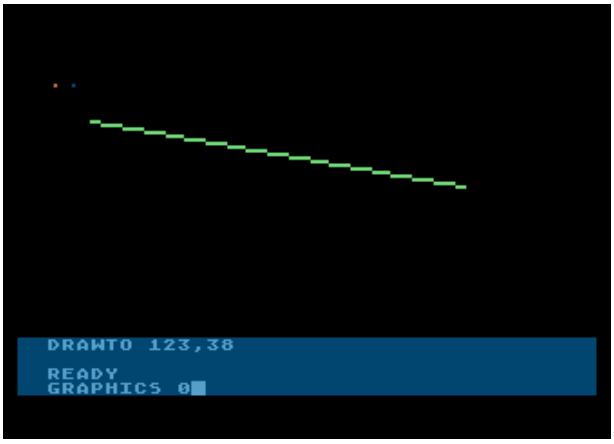
# BASIC Dili

Atari 800 serisinde gelen basic dili oldukça gelişmiş bir basic dilidir. Bu dili en güçlü kılan özellik donanım özelliklerinin birçoğuna rahatça erişebilmesidir. Özellikle grafik modları ve grafik komutları açısından zengin bir dildir. Ayrıca bir grafik modu açıkken satır satır program çalıştırmak için de özel tasarlanmıştır. Atari Basic'i, birçok grafik modu için ekranın alt bölümünde bir text alanı ayırır ve imleç bu alanın sınırlarından dışarı çıkamaz. Bu sayede ekranın üst tarafında bir grafik varken alt tarafta hala basic komutlarını görmek mümkündür. Bununla ilgili hemen bir örnek görelim.

Aşağıdaki satırları Atari 800 XL/XE modellerinin açılış ekranlarında sırayla yazıp Return (Enter) tuşuna basın.

```
graphics 7
color 1
plot 10,10
color 3
plot 15,10
color 2
plot 20,20
drawto 123,38
graphics 0
```

Bu komutları girdikçe yukarıda anlatılmak istenen kullanışlılığı göreceksiniz. Elbette ki bu grafik modları tam ekran olarak da basicden kullanılabilirler.




Komutların çıktısı bu şekilde olacaktır.

**Şekil 7.**

# Atari'de Makine Dilinden Program Yazmak

Atari basici ne kadar gelişmiş olursa olsun, 2 Mhz'in altındaki bir bilgisayar söz konusu olduğunda makine dili kullanımı kaçınılmaz bir hal alıyor. Atari'de makine dilinden program yazabilmek için sahip olunması gereken temel bilgi, 6502 assembler programlama bilgisidir. Bu temele sahip olunması durumunda aşılması gereken iki engel daha vardır.

1. Assembler programı yazabilecek rahat bir ortam oluşturmak.
2. Atari'nin hazıfa yapısı ve donanımsal özelliklerini öğrenmek.

İlk madde için ihtiyacımız olan şey assemblerdan programlarımızı yazabileceğimiz bir editör/derleyici. Atari üzerinde bu güne kadar bu amaçla en sık kullanılmış derleyici "Atari Assembler Editor" ya da kısaltılmış ismiyle "Atari ASM Edit" kartuşudur. 8 KB'lık bir ROM'a sahip olan kartuş, zamanına göre çok üstün özelliklere sahiptir. Özellikle program içersinde hata yakalamak için (debugging) çok kullanışlı fonksiyonları vardır. Tek eksiği 8 KB'lık hafızaya sığdıramadıkları için hata mesajlarının yalnızca sayısal kodlardan ibaret olmasıdır. Kısacası bir hata aldığınızda ekranda hatanın tanımını değil yalnızca kodunu görürsünüz ve kartuşun dökümanından bu hata kodunun ne anlama geldiğini araştırmanız gerekir. Bu kartuşu kullanmak isterseniz temel kullanım bilgileri aşağıda örneklerle verilmiştir.

# Atari Assembler Editor Kartuşu Kullanım Örneği

Kartuşu taktığınızda "EDIT" yazan bir ekran karşınıza gelecek. Bu ekranda Basic'den program yazar gibi satır numaralarıyla program yazabilirsiniz. Dikkat etmeniz gereken husus, satır numarasından sonra 1 değil en az 2 boşluk bırakmanız gerektiğidir. Örnek program:

```
10  *= $0600
20  LDA #$C4
30  STA $02C6
40  RTS
```

Programımızı yazdıktan sonra "LIST" komutuyla programın listesini alabilir ve istediğiniz taktirde "NEW" komutuyla listeyi temizleyebilirsiniz. Programı yazdık ancak program henüz derlenmedi. Programı derlemek için "ASM" komutunu kullanmamız gerekiyor. ASM komutunu yazıp "Return" tuşuna bastığımızda aşağıdaki gibi bir çıktı almamız lazım.

```
ASM
0000          10          *=    $0600

0600 A9C4     20          LDA   #$C4

0602 8DC602 30            STA   $02C6

0605 60       40          RTS
```

Eğer aşağıdaki çıktıyı aldıysak programı düzgün bir biçimde derlemeyi becermişiz demektir. Daha sonra programı çalıştırmak ve olası hataları takip etmek için "Edit" bölümünden "Debug" bölümüne geçiş yapmamız gerekiyor. Bunun için kullanacağımız komut "BUG" komutu. Programı çalıştırmak için ise go (git) anlamına gelen "G" harfini kullanacağız. Bu harfin yanına programın başlangıç adresini hexadecimal olarak yazmamız gerekiyor. Kısacası bundan sonra yazmamız gereken komutlar ve ekranda göreceğimiz çıktılar şu şekilde olmalı;

```
BUG

DEBUG
G 0600
0003        A=C4 X=00 Y=00 P=B0 S=12
```

Bu çıktının yanısıra ekran renginin yeşile döndüğünü herhalde fark etmiş olmanız lazım. İşte minik programımızın yaptığı iş de aynen bu. $02C6 hafıza adresi arkaplan renginin tutulduğu adres. $C4 yerine farklı değerler yazarak Atari'nin güzel renk paletini incelemeniz mümkündür.

Son olarak "Debug" moddan çıkıp "Edit" moduna geri dönmek için "X" komutunu kullanıyoruz. Örnek:

```
X

EDIT
```

Bundan sonra programı değiştirmeye devam edebilir ya da yeni bir program yazmaya girişebilirsiniz. Basic dilinde olduğu gibi aynı satır numarasını tekrar yazıp "Return" tuşuna basarsanız, önceki satır yenisiyle değiştirilecek, yeni satır eskisinin üzerine yazılacaktır.


Programın sonucu Atari'de bu şekilde görünecektir.

**Şekil 8.**

# Atari Assembler Editor Kartuşu Hata Kodu Listesi

Programlarınızı derlerken alabileceğiniz hata kodlarının anlamları aşağıdaki listede orijinal dokümanından alındığı şekliyle verilmiştir.

1 : Insufficient memory for assembly

2 : The number xx cannot be found for the "DEL xx,yy" command

3 : Error in specifying an address in mini-assembler

4 : File cannot be loaded

5 : Undefined reference label

6 : Syntax error in a statement

7 : Label defined more than once

8 : Buffer overflow

9 : No label given before "=".

10 : Byte expression is greater than 255

11 : Null string used where invalid

12 : Address or address type specified is incorrect

13 : Phase error: inconsistent result found from pass 1 to pass 2

14 : Undefined forward reference

15 : Line too large

16 : Source statement not recognized by assembler

17 : Line number too large

18 : LOMEM command attempted after other commands/instructions

19 : No starting address given

# Atari Assembler Editor İle İlgili Detaylı Bilgiler

Program yazmayı, çalıştırmayı öğrendik. Ancak Atari Assembler Editor'ün yetenekleri çok fazla. Eğer gerçekten bu kartuş hoşunuza gider ve kullanmak isterseniz daha fazla detay için "www.atariarchives.org" sitesini gezmenizi öneririm. Özellikle aşağıdaki adreste ihtiyacınız olan diğer komutların bir kısmını bulabilirsiniz.

http://www.atariarchives.org/roots/chapter_5.php

Biz şimdi bu kartuşu kullanmak yerine farklı bir yol çizeceğiz.

# Atari İçin Çapraz Geliştirme

Atari Assembler Editor'ün nasıl kullanıldığını gördük. Ancak günümüzde Atari üzerinde çalışmak o kadar da pratik bir yol değil. Birçoğumuzun elinde çalışır halde bir Atari bile yok (ben de dahil olmak üzere). Olanların da çalışmalarını kaset ya da 5.25 disketlere emanet etmek isteyeceklerinden şüpheliyim. Dolayısıyla nostaljik sebepleri bir kenara bırakacak olursak PC ve emülatör kullanarak Atari için program yazmak en güvenilir ve rahat çözüm olacaktır.

Atari için uzun süre çapraz geliştirme ortamımı hazırlamak için araştırmalar yaptım. Bazı açık kaynak kodlu amatörce geliştirilen projelerde kullanılan çapraz geliştirme araçlarını gördüm, inceledim. Ancak dönüp dolaşıp eski bir dost olan ACME Crossassembler'da karar kıldım. Aslında ACME genellikle Commodore 64'e yönelik olarak hazırlanmış bir derleyici. Ancak sonuç olarak 6502 her halikarda 6502'dir. Yani ACME'nin üreteceği binary kodlar Atari'de çalışabilecek durumda olacaklar. Peki bu durumda herşey çözülmüş oluyor mu? Ne yazık ki hayır, biraz daha çaba sarf edeceğiz.

Problemlerimiz neler? ACME'nin üreteceği binary dosyaları Atari'nin hafızasına istediğimiz adresten yükletmek ve çalıştırmak zahmetli bir iş olacaktır. Commodore 64'ün *.prg dosya formatında dosyanın ilk iki byte'ı 16 bitlik bir hafıza adresi içerir ve bu sayede Commodore 64 dosyayı hafızaya nereden yükleyeceğini bilir. Ancak Atari'de durumlar biraz farklı.

Araştırmalarım sonunda Atari'nin otomatik olarak program yüklendiği gibi çalışabilen bir EXE formatını buldum. PC EXE'leriyle karıştırılmaması için genellikle XEX şeklinde uzantı kullanıyorlar. Bu dosyalar kolaylıkla yüklenebiliyor ve çalıştırılabiliyorlar.

# XEX Dosya Formatı

Bu dosya formatı programın başlangıç ve bitiş adreslerine ihtiyaç duyar. En temel haliyle format şu şekilde verilebilir.

Kullanacağımız kısaltmalar:

BA : Başlangıç Adresi

BT : Bitiş Adresi

XEX Dosya Yapısı:

2 byte : $ffff (dosya başlığı)

2 byte : BA (16 bit)

2 byte : BT (16 bit)

N byte : Programı bu bölüme yazıyoruz

2 byte : $02e0

2 byte : $02e1

2 byte : BA (16 bit)

Eğer byteları tek tek sıralayacak olursak.

$ff, $ff, BA(alt 8 bit), BA(üst 8 bit), BT(alt 8 bit), BT(üst 8 bit), ... , $e0, $02, $e1, $02, BA(alt 8 bit), BA(üst 8 bit)

Böyle bir byte dizilimiyle karşılaşıyoruz.

# ACME Kullanarak XEX Dosyası Yaratmak

Şimdi yukarıda verdiğimiz dosyanın ACME'den nasıl oluşturulduğunu görelim. Atari Assembler Editor için yaptığımız örneğin aynısını ACME'den kullanarak hazırlayalım.

renk.ata

```
BASLANGIC_ADRESI = $0600

!to "renk.xex",plain

* = BASLANGIC_ADRESI-6
!word $ffff, BASLANGIC_ADRESI, BITIS_ADRESI

* = BASLANGIC_ADRESI
lda #$c4
sta $02c6
jmp $a000

BITIS_ADRESI = *
!word $02e0, $02e1, BASLANGIC_ADRESI
```

Bu örneğin tek farkı sonunda "RTS" yerin" JMP $a000" kullanmamız. Bunun nedeni ise otomatik olarak çalıştırılan programlar Basic'den çağırılmıyorlar. Bu durumda da geri döneceğimiz nokta Basic olmadığı için programdan çıktığımız takdirde bilgisayar reset atıyor ve bunun gibi bir örnekte sonucu göremiyebiliyoruz. Basic'e dönmeden sonucu görmek istersek "RTS" yerine "JMP *" kullanabiliriz. Bu durumda programımız çalışacak ve sonunda sonsuz döngüye girerek kilitlenecektir. Ancak ekran renginin değiştiğini gözlemleyebiliriz. Atari'de Basic'in başlangıç adresi $a000'dır. "jmp *" yerine "jmp $a000" kullandığımız durumda programımız çalışıyor ve basic ekranına güvenli bir şekilde ulaşmış oluyoruz.

Şimdi yukarıdaki örneği inceleyelim. İlk olarak BASLANGIC_ADRESI isimli bir sabit tanımlıyoruz. $0600 rastlantısal bir adres değil. Otomatik olarak çalıştıracağımız programlarımızı bu adrese yerleştirmemiz önemli. Tabii ki istediğimiz taktirde programımız farklı bir hafıza adresine de yerleşebiliyor. Fakat yine de ilk olarak başlangıç adresinin $0600 verilmesi gerekiyor. Daha sonra program istenilen adrese dallanabiliyor.

!to "renk.xex",plain

Bu satırda dikkat etmemiz gereken noktalar şunlar. "renk.xex" derlenmiş programın kaydedileceği dosya ismi. Ancak asıl önemli olan ondan sonraki "plain" parametresi. Eğer ACME'yi Commodore 64'e yönelik kullanıyor olsaydık buraya "cbm" yazardık. Bu sayede otomatik olarak başlangıç adresi programın başına eklenirdi. Kısacası ACME'nin Atari'ye özel bir dosya formatı desteği olsaydı buraya "ATR" gibi birşey yazıp doğrudan programımızı yazmaya geçebilirdik. Ancak ne yazık ki ACME'nin henüz Atari'ye özel bir desteği bulunmuyor. Ama ACME'nin yapımcıları, ACME'nin farklı 6502 tabanlı platformlarda kullanılabileceğini öngördükleri için "plain" parametresini eklemeyi ihmal etmemişler. Bu parametrenin yaptığı şey ise dosyanın başına, sonuna hiçbirşey eklememek ve düz bir binary dosya oluşturmak. Biz bu özelliği kullanarak kendi istediğimiz bir dosya formatını oluşturabiliyoruz.

* = BASLANGIC_ADRESI-6

XEX dosyalarının 6 bytelık bir başlığı (header) bulunuyor. Biz de burada başlangıç adresinin 6 byte öncesine denk gelecek şekilde başlık için pozisyon ayarlaması yapıyoruz.

!word $ffff, BASLANGIC_ADRESI, BITIS_ADRESI

Bu satır dosyanın başlığını oluşturuyor. !word 16 bitlik değerleri doğrudan yazmamızı sağlıyor. Eğer byte byte yazacak olsaydık bu satırı şu şekilde kullanabilirdik. !byte $ff, $ff, <BASLANGIC_ADRESI, >BASLANGIC_ADRESI, <BITIS_ADRESI, >BITIS_ADRESI

* = BASLANGIC_ADRESI

Programın başlangıcına geldik. Aslında bu satırı yazmasak da tam 6 bytelık bir başlık tanımladığımız durumda aynı noktada olmamız gerekiyor. Ancak hem programın daha rahat okunması için hem de bir tedbir olarak bu satırı ekledik.

BITIS_ADRESI = *

Programın bitiş adresini baştan tanımlamak mantıklı bir yol değildir. Çünkü program uzayıp kısaldıkça bitiş adresi değişecektir. Her defasında bu adresi bulmaya uğraşmak da çok mantıklı bir seçenek değildir. Bu yüzden programın sonuna geldiğimizde "*" sembolünü kullanarak program adres sayacının değerini BITIS_ADRESI'ne eşitliyoruz. Bu da doğrudan olarak başlıktaki yerini alıyor.

!word $02e0, $02e1, BASLANGIC_ADRESI

$02e0 ve $02e1 adresleri Atari'nin disk işletim sistemi tarafından kullanılan adresler. Programın sonuna eklediğimiz BASLANGIC_ADRESI'de yine programın çalıştırılacağı başlangıç adresinin yer aldığı yer.

# Merhaba Dünya Örneği

Elbette ki bu örneği unutmuş değiliz. Atari'de ekrana yazı yazdırmanın birkaç farklı yolu var. Şimdilik diğer hafıza adreslerine çok bulaşmadan yalnızca ekran adreslerine kopyalama yaparak nasıl ekrana yazı yazdırıldığını görelim.

merhaba_dunya.ata

```
BASLANGIC_ADRESI = $0600
EKRAN = $9c40

!to "merhabadunya.xex",plain

* = BASLANGIC_ADRESI-6
!word $ffff, BASLANGIC_ADRESI, BITIS_ADRESI

* = BASLANGIC_ADRESI
ldx #$00
dongu
lda YAZI,x
sta EKRAN+82,x
inx
cpx #YAZI_SONU-YAZI
bne dongu
jmp $a000

YAZI !text "merhaba",0,"dunya"
YAZI_SONU = *

BITIS_ADRESI = *
!word $02e0, $02e1, BASLANGIC_ADRESI
```

Atari'nin text ekranının başlangıç adresi $9c40'dır. Programın başında EKRAN sabitine bu değeri veriyoruz. Daha sonra bir döngü ile YAZI'da tanımladığımız karakterleri ekrana basıyoruz. Toplamda kaç karakterin kopyalanacağını YAZI'nın bitimine koyduğumuz YAZI_SONU adresiyle YAZI'nın farkını alarak bulabiliyoruz.

EKRAN'ın başlangıcı basic'de imleçin dolaşabildiği alanın biraz dışında kalıyor. EKRAN'a 82 ekleyerek basic'de "READY" yazan satırın tam altına yazıyı yazdırmış oluyoruz.


**Şekil 9.**

Son olarak dikkat edilmesi gereken nokta "!text"in Atari'nin ASCII karakter kodlarına uygun sonuç üretmediğidir. Küçük harfler söz konusu olduğunda bir problem yaşamazsak da diğer birçok karakter (boşluk karakteri de dahil olmak üzere) farklı sonuçlar verecektir. Bu yüzden, gerektiği yerde, karakter kodlarının kullanılması gerekiyor. Örneğin boşluk karakteri !text ile yazdırıldığında $20 (32) karakter kodunu oluşturur. Bu Commodore 64'de boşluğa denk gelirken Atari'de @ karakterine denk gelir. Atari'de boşluk karakteri 0 kodundadır. Bunun için "merhaba dunya" yerine "merhaba",0,"dunya" yazmamız gerekiyor.

# Assembler'dan Grafik Tabanlı Programlar Yazmak

Atari'de Assembler'dan grafik ekranı açmak oldukça kısa bir program parçasıyla yapılabiliyor. Örnek vermek gerekirse;

```
LDA #$00
STA $022f
LDA #<GORUNTU_LISTESI
STA $0230
LDA #>GORUNTU_LISTESI
STA $0231
LDA #$22
STA $022f
```

bu program parçası istenilen grafik modunu açabilir. $022f adresiyle Antic çipini yönetebiliyoruz. Bu adrese 0 değeri verdiğimizde Antic çipi kapanıyor. Böylece Antic çipinin okuduğu görüntü listesini (display list) değiştirme imkanı elde ediyoruz. $0230 ve $0231 adresleri bu listenin işaretçisi. $0230'a listenin yer aldığı hafıza adresinin alt byteını, $0231'e ise üst byteını koyuyoruz. Örneğin görüntü listemiz $2560 adresinde yer alıyor ise $0230'a $60, $0231'e $25 değeri vermemiz gerekiyor. Son olarak $022f adresine $22 değerini yazarak yeniden Antic çipini aktif ediyoruz ve yeni ekran görüntümüzle karşılaşıyoruz.

Buradaki soruişaretleri şunlar. $00, $22 gibi değerler nereden geliyor? Görüntü listesi tam olarak nasıl birşey?

$022f adresi tanımlamalarda genellikle SDMCTL şeklinde kısaltılmıştır. Anlamı DMA kontrol registerını kaydet demektir. Bu adrese yazılan değerler birer komut olarak algılanır. Yani $00 - Antic çipini kapat anlamına gelirken $22 - Antic çipini aktive et anlamına gelir.

# Görüntü Listesi (Display List)

Görüntü listesi komutlar ve bu komutların parametrelerinden oluşan bir listedir. Bu yönüyle Amiga'nın Copper'ına benzetilebilir. Yukarıdaki program parçasında hangi grafik modunun açılacağı belirtilmemiştir. Bu tamamen görüntü listesinde yer alan değerlerle belirlenir.

Görüntü listelerinde kullanılabilecek komutlar aşağıdaki gibidir.

Boş Tarama Satırı Kodları:

$00 : 1 boş satır

$10 : 2 boş satır

$20 : 3 boş satır

$30 : 4 boş satır

$40 : 5 boş satır

$50 : 6 boş satır

$60 : 7 boş satır

$70 : 8 boş satır

Görüntü Kodları:

$02 : Text Mod 0

$03 : Text Mod *

$04 : Text Mod *

$05 : Text Mod *

$06 : Text Mod 1

$07 : Text Mod 2

$08 : Grafik Mod 3

$09 : Grafik Mod 4

$0A : Grafik Mod 5

$0B : Grafik Mod 6

$0C : Grafik Mod *

$0D : Grafik Mod 7

$0E : Grafik Mod *

$0F : Grafik Mod 8

(*) ile işaretli modlar basicden kullanılamayan modlardır.

Sıçrama Kodları:

$01 : Adrese sıçra

$41 : Adrese sıçra ve taramayı yakalamayı bekle

Örneğin;

* listeye $06 değeri yazılarak ANTIC 6 (bkz: Tablo 1) moduna geçilebilir.

* 8 satır, yani 1 normal karakter yüksekliği boşluk vermek için $70 değeri kullanılabilir.

* $01 komutu ile liste içersinden bir hafıza adresine sıçramak ve listeye oradan devam etmek mümkündür. $41 komutu ise sıçrama ve tarama yakalanana kadar bekleme anlamlarına gelmektedir. $01 ya da $41'den sonra gelen 16 bitlik adres, sıçranacak hafıza adresidir. Listenin başlangıç adresinin verilmesi durumunda sabit olarak o listeye uygun olan grafik ekranı açık kalacaktır.

Görüntü listesinin özellikleri bunlarla da bitmez. Görüntü kodları yalnızca 0,1,2 ve 3 numaralı bitler kullanılacak verilebilmektedir. 4,5,6 ve 7 numaralı bitler ise farklı özellikleri devreye sokarlar. Yani;

%XXXXYYYY

YYYY : Görüntü kodu

XXXX : Ek parametreler

şeklinde düşünülebilir. Bu parametreler şunlardır:

Bit 4: Dikey Kayma (Vertical Scroll). [Görütü kodu+$10]

Bit 5: Yatay Kayma (Horzontal Scroll). [Görütü kodu+$20]

Bit 6: Hafıza Taraması Yükleme (Load Memory Scan). [Görütü kodu+$40]

Bit 7: Görüntü Listesi Kesmesi (Display List Interrupt). [Görütü kodu+$80]

Eğer 6. biti yani "Hafıza Taraması Yükeleme"yi kullanacak olursak görüntü listesinde bu görüntü kodunun ardından gelen iki byte ekran adresinin başlangıcına işaret eder. Yani görüntü kodunu $06 yerine $06+$40 = $46 şeklinde kullanacak olursak yine ANTIC 6 grafik modu açmış oluruz ancak bu defa sonrasında gelen 2 byte'ın değeri ile ekran adresinin farklı bir adrese kaydırılması sağlanmış olur. Bu işlem farklı satırlar için defalarca tekrarlanabilir.

# Grafik Örneği

Şimdi bu öğrendiklerimizi bir araya getiren bir örnek görelim.

grafik.ata

```
BASLANGIC_ADRESI = $0600

SDMCTL=$022F
SDLSTL=$0230
SDLSTH=$0231
RENK0=$02C4
RENK1=$02C5
RENK2=$02C6
RENK3=$02C7
RENK4=$02C8

!to "grafik.xex",plain

* = BASLANGIC_ADRESI-6
!word $ffff, BASLANGIC_ADRESI, BITIS_ADRESI

* = BASLANGIC_ADRESI
; Renkleri Ayarla
lda RENK3
sta RENK1
lda RENK4
sta RENK2
; Grafik Ekranini Ac
lda #0
sta SDMCTL
lda #<GORUNTU_LISTESI
sta SDLSTL
lda #>GORUNTU_LISTESI
sta SDLSTH
lda #$22
sta SDMCTL
```

```
jmp $a000

; Yazilar
SATIR1 !TEXT 0,0,0,0,0,0,"satir",0,"bir",0,0,0,0,0
SATIR2 !TEXT 0,0,0,0,0,0,"satir",0,"iki",0,0,0,0,0
SATIR3 !TEXT 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
       !TEXT "satir",0,0,"uc",0,0,0,0,0,0,0,0,0,0
SATIR4 !TEXT 0,0,0,0,0,0,"satir",0,"dort",0,0,0,0

; Goruntu Listesi
GORUNTU_LISTESI
; Bos Satirlar
!byte $70,$70,$70,$70,$70,$70,$70,$70
; Antic 6 Grafik Modu + Hafiza Taramasi Yukleme
!byte $46
; Ekran Hafizasini SATIR1'e Ayarla
!word SATIR1
; Bos Satirlar
!byte $70,$70,$70,$70
; Antic 7 Grafik Modu + Hafiza Taramasi Yukleme
!byte $47
; Ekran Hafizasini SATIR2'ye Ayarla
!word SATIR2
; Bos Satir
!byte $70
; Antic 2 Grafik Modu + Hafiza Taramasi Yukleme
!byte $42
; Ekran Hafizasini SATIR3'e Ayarla
!word SATIR3
; Bos Satirlar
!byte $70,$70,$70,$70
; Antic 6 Grafik Modu + Hafiza Taramasi Yukleme
!byte $46
; Ekran Hafizasini SATIR4'e Ayarla
!word SATIR4
; Bos Satirlar
!byte $70,$70,$70,$70,$70
; Adrese Sicra ve
; Tarama Yakalanana Kadar Bekle
!byte $41
; Sicranacak Adres
!word GORUNTU_LISTESI

BITIS_ADRESI = *
!word $02e0, $02e1, BASLANGIC_ADRESI
```

Bu örnek ANTIC 6, ANTIC 7, ANTIC 2 ve tekrar ANTIC 6 grafik modu açarak her bir modda birer satır yazı yazar. Ancak bu yazı yazma işlemi ekrana bir çıktı vererek değil hafızada yazıların bulunduğu adresler ekran adresi olarak değiştirilerek yapılır. Kısacası ANTIC çipinin ve bu çipin bize sağladığı görüntü listesi özelliğinin gücünden yararlanır.


**Şekil 10.**

# Değerlendirme

Atari ile ilgili anlatılacak daha çok fazla konu var. Araştırmalarım sonunda Atari'nin yapabilecekleri, donanımını kullanma gücü ve sınırları ile ilgili çok ilginç bilgiler edindim. Elbette ki ister istemez benim 20 yıllık aşkım olan ve yine 6502 mimarisinden gelen Commodore 64 ile de sık sık mukayese ettim. İlginç olan şudur ki bir iki konu haricinde Atari 800 XL/XE modelleri Commodore 64'ü ezip geçiyor. Atari'nin piyasadan silinmesinin ardından Commodore 64'ün uzun yıllar varlığını sürdürebilmesi ve Atari'ye göre satış rakamlarındaki büyük fark, Commodore firmasının başarısını kanıtlayan öğelerden biri. Bunca yıl boyunca kandırılmışız. Atari Commodore'dan daha üstün bir bilgisayarmış (her konuda olmasa da). :)

Aslında bu yazıyı bir yazı serisi olarak tasarlamamıştım. Yazının çıkış noktası ACME kullanarak Atari üzerinde nasıl çapraz geliştirme yapılabileceğini anlatmaktı. Ama yazıyı yazmaya gelince işler bazen değişiyor. Beklediğimden çok daha fazla detaya girmek durumunda kaldım. Bu da bir bakıma bu yazının devamı olacağını gösteriyor.

Daha anlatılacak neler mi var? Öncelikle donanımda gelen "Oyuncular ve Füzeler" konusundan bahsetmek çok eğlenceli olacaktır. Böyle bir başlık atmak bile başlı başına bir zevk :) Ayrıca biraz daha intro/demoya yönelik olarak kayan yazılar, grafik ekranına şekiller çizdirme, renkler ve parlaklık değerleri ile yapılabilecek efektler gibi konular işin olmazsa olmazları gibi duruyor. Tabii ki kesme rutinleri, ses özellikleri ve diğer birçok konu bu yazının birden bile fazla devam yazısı olabileceğini gösteriyor.

# Kaynaklar

Bu yazıyı hazırlarken birçok kaynaktan yararlandım. Ne de olsa bu platformla en son uğraştığımda 8-9 yaşlarındaydım. O zamanlar öğrendiğim 3-5 basic komutunun üzerine epey bir bilgi eklemem gerekti haliyle. Bu yazıyı hazırlamadan önceki birkaç aylık öğrenme sürecimi de hesaba katacak olursak, tek tek faydalandığım bütün kaynakların listesini veremesem de temel olarak kullandığım iki kaynağı belirtmek isterim:

Atari Archives:

http://www.atariarchives.org/

Wikipedia'da Atari 800:

http://en.wikipedia.org/wiki/Atari_800

Atari Archives çok geniş bir arşiv. Giriş sayfasındaki görüntü ilk bakışta içeriksiz bir reklam sitesi gibi dursa da aşağıdaki banner benzeri linklere tıkladıkça o linklerin hepsinin altında büyük birer bilgi hazinesi yattığını görüyorsunuz. İngilizce okuma düzeyi yeterli olan herkese tavsiye ederim.

Bir sonraki sayıda 2. bölümde görüşmek dileğiyle, hoşçakalın. [televizyonvari bir bitiriş oldu... ;)]

emir (at) akaydin (nokta) com

# Bir Scener Eşinin Gözünden Demoscene ve Scenerlar

Seval (Irian) Çakır

**Şekil 1.**

Herkese Merhabalar,

Ben Irian, bu derginin editörü Nightlord'un eşi. Nightshift 2006'ya iştirakimden dolayı benim de bir nickname'im var ama onu kullanmak ve tanıdığım scenerlara nickname'leri ile hitap etmek, yıllardır alışamadığım bir şey (Vigo hariç, nedense ona Uğur demek daha tuhaf geliyor :) )

Utanarak itiraf ediyorum, bu yazının Plazma'nın bir önceki sayısında yer alması gerekiyordu. Sanırım Nightlord'un zamanında yazısını teslim alamadığı tek yazar ben oldum. Ama şunu da belirtmeliyim ki, dergiye akan yazıları gördükten sonra, bana deadline'ı ikinci kere hatırlatmadı bile :) Fakat Plazma 4'ün yayınlandığı gün, derginin ilk 10 sayfasını okuduktan sonra öyle gaza geldim ki, gece 3'te kalkıp haftalardır elimde sürünen yazının ilk halini bitirdim. Ayrıca belirtmeden geçemeyeceğim, gördüğüm ilk andan itibaren kapağa bakmaya doyamadım. Çok çarpıcı ve çok profesyonel… Onun da etkisi olmadı değil.

Ocak 2008 itibariyle Nightlord'la 10. yılımızı doldurmuş bulunuyoruz. Bu yazıda, 10 yılda benim demoscene kavramım nerden nereye geldi açıklamaya çalışacağım, çünkü Nightlord bunun faydalı olabileceğini düşünüyor :) Umarım farklı bir bakış açısı okuyan herkesin hoşuna gider.

2. ayımızın içerisindeyken (o zaman üniversite 2. sınıftayız) Nightlord u haftada bir veya iki defa görüyorum. Biliyorum ki spor yapıyor, dolayısı ile haftada en az bir gününü antrenman yaparak geçiriyor. Onun dışında okula çok gelmiyor, derslere girmiyor, bir zahmet sınavlara giriyor. Ve sınavların sonuçlarından anlaşıldığı kadarı ile, boş vakitlerinde pek ders çalışmıyor. Peki bu adam ne yapıyor? Diye merak etmeye başlıyorum, ama sormuyorum.

4. aya doğru bu durum pek gelişme göstermeyince, dayanamayıp soruyorum. O da yeni bilgisayar aldığını ve sürekli onunla uğraştığını söylüyor. Tabi bu bana pek bir şey ifade etmiyor. Neden? Çünkü bilgisayarın başında oyun oynamak haricinde (o zaman daha internet evlerimize girmiş değil :) ) bu kadar çok vakit geçirilip de ne yapılır, hiç bir fikrim yok. İçinde bulunduğum fikirsizliği daha iyi anlayabilmeniz için benim o güne kadar bilgisayarla olan ilişkimi biraz açmak durumundayım.

12 yaş civarlarındayken, benimle yaşıt kuzenime Commodore alınıyor. Dayım nerden duydu aldı bilmiyorum, ama bir şekilde alıyor sonuçta. Fakat 1) ayrı şehirlerde yaşadığımız için çok az görüşebiliyoruz. 2) derslerimizi kötü etkilemesi olasılığına karşılık, görüştüğümüz o kısıtlı zamanda da oynamamıza çok az izin veriliyor. Oynamamıza diyorum çünkü başka bir şey yapma aşamasına gelemeden Commodore evden gönderiliyor. Dolayısı ile benim Commodore 64 ile tanışıklığım toplam 45 dakikalık bir temastan ibaret. Ve benim için sadece bir oyun konsolu, başka özelliklerinin olduğunu çok çok uzun yıllar sonra öğreniyorum.

Herhangi bir bilgisayarla ikinci temasım, lisede aldığım bilgisayar dersi. Ama durum şöyle, toplam 10 bilgisayar var, biz 50 kişiyiz. Dolayısı ile 5 kişi sıra ile kullanmak durumundayız. Kullanmak derken, ne yapabileceğimiz hakkında bir fikrimiz yok, cesaretimiz de yok. Bozarız korkusu ile bakıyoruz kendisine. Aksi gibi öğretmenimizin de pek fikri yok. Bu bilgisayar modası yeni çıkmış durumda, adamcağıza aldırmışlar 3-5 saat ders bir yerlerden, kitaptan bakıp bakıp yazdırıyor bize. Güya basicte bir şeyler yazmayı öğreniyoruz. 2+2'nin sonucunu öğrenmek için neden bu kadar uğraştığımızı sorgulayamadan dönem bitiyor.

Ben bu durumda, bir Psikoloji öğrencisi olarak üniversiteye başlayıp, yurdun laboratuarında düşe kalka, word ile ödevimi yazıp, diskete kaydetmeyi öğreniyorum.

İşte Nightlord haftanın 5 gününü bilgisayar başında uğraşarak geçirdiğini söylerken ben bu durumdayım. (Fark ettiğiniz üzere, benim için henüz bir sohbette adı bile geçmedi demoscene kavramının). Kendisine pek inanmıyorum, ama devam ediyorum.

Birinci yılımıza yaklaşırken, Nightlord un bu kadar vakti hakikaten bilgisayarın başında geçirdiğine ikna oluyorum. Demek ki, bu da böyle birisi deyip kabulleniyorum. O yaz bana PC de yaptığı bir müziği dinletiyor. Çok beğenip, çok şaşırıyorum. O zaman bana kullandığı müzik programını gösteriyor. "wow"

Yıllar geçiyor evleniyoruz. İlk bir yıl sürekli şu moddaydım: "Allah'ım ben sürekli bilgisayar başındaki bu adamın sırtını seyretmek durumunda mıyım? Ne zaman sıkılacak da dışarı çıkmak isteyecek?" Ama onun canı asla sıkılmıyor, her zaman yapacak bir şeyleri var. Tam aksine o aralar, yıllardır inaktif olduğuna inandığı eski bir hobisinin Türkiye'de hala aktif olduğunu

keşfediyor. Ve kabusum başlıyor “Allah aşkına nedir bu demoscene”

Bunlar da yetmiyormuş gibi, tam o sırada ailesi yıllardır oturduğu evden taşındığı için, eski odasındaki her şeyi bize gönderiyorlar. Yarabbi kutular dolusu teyp kaseti (sonra onların Commodore oyun kasetleri olduğunu öğreniyorum), dergiler, Commodore disketleri, bilgisayar parçaları, daha ne olduğu konusunda hiçbir fikrimin olmadığı binlerce ıvır zıvır. Nightlord bunları kutsal emanetlermiş gibi karşılayıp, evimize gelen misafire yatak odası olmasını planladığımız odayı bir güzel donatıyor. Neyse !

İlk birkaç hafta, aralıksız demo seyrediyor. Çok sevdiklerini üst üste defalarca seyrediyor. Odasından gelen demo müziği, Çin işkencesi gibi, pencereyi açıp “İmdat” diye bağırmak istiyorum.

Bu on yılın içindeki en zor zamanlar sanırım bu keşfin hemen arkasından, ilk demosunu yapmaya karar verdiği zamanlardır. Partiye yetiştirme telaşı içinde kendini o kadar kaptırdı ki, birkaç ay görüşmedik desem yeridir. O sure içerisinde o kadar çok şikayet edip, söylendim ki ben bile kendimden nefret ettim, o hiç etkilenmeyip azimle devam etti :) O kadar kızdım ki, ne yaptığını anlamak için hiç çaba göstermedim. Sonuçta okumanın haricinde herhangi bir hobisi olmayan sıradan bir Türk insanıydım ben ve ufkum almıyordu bu olayı.

Ta ki bu birkaç ayın sonunda, Nightlord beni PC’nin başına oturtup ilk demosunu seyrettirene kadar. Water’ı hayranlık ve şaşkınlık içinde seyrettim. Bu kadar zaman uğraşıp bu çok etkileyici ama kısacık şeyi mi yapmıştı. Demek bu kadar zor bir olaydı bu. Ve o kadar heyecanlı ve mutluydu ki. Sanırım bir şeyler üretmenin çok farklı ve insanı mutluluktan deli edebilecek bir olay olduğunu ilk o zaman fark ettim. Benim için o demo dönüm noktasıdır. Kızmayı bırakıp, ne yapıyor diye ilgilenmeye başlamamın dönüm noktası. (bu demonun bana ithaf edilmiş olmasının bu kararımda sanıldığı kadar etkisi yoktur :) )

İtiraf etmeliyim ki bu ikimize de iyi geldi. Benim açımdan, ilk olarak söylenmeyi bırakmış oldum (ya da çok azalttım diyelim). Dolayısı ile huzurum(uz) arttı. İkincisi, sırtını seyretmeyi bırakıp, omzunun üstünden ekranı seyretmeye başladım ki, gördüklerimi beğendiğimi itiraf etmeliyim. Artık ciddi bir demo izleyicisiyim. Tabii ki kendim takip edip, indirip izlemiyorum. Ama Nightlord izlerken ona katılmak hoşuma gidiyor. Hatta favori gruplarım ve demolarım bile var (Glance/Living mesela :) ) Pixel Art a olan saygım Hydrogen in Bronx logosunu yapmaya çalışma teşebbüsümle bin kat arttı. Baka baka taklit bile edemedim :) Artık her grafiğin önünde saygıyla eğiliyorum. Müziğe hayatımın her aşamasında çok ilgisiz biriydim (çok yetenekli bir ailenin yeteneksiz tek ferdi olmanın verdiği utançla sanırım) ama pencereyi açıp imdat diye bağırma aşamasından, bazı demo müziklerini MP3 player’ıma yükleyip dinleme aşamasına geldim. Gösterdiğim gelişmeyi takdir etmelisiniz :)

Tabi bunları, kendi gösterdiğim çaba ve sabrın haricinde sevgili öğretmenim Nightlord a da borçluyum. Yıllar boyunca bütün sorularıma hiç sıkılmadan cevap verip her şeyi benim anlayabileceğim seviyede anlattı. Aynı demoyu belki onuncu kez seyredip, onuncu kere hangi gruba aitti diye sorsam bile :) (Bütün bunları camiaya bir scener daha kazandırabilir miyim diye yaptığından şüphelenmiyor değilim:)) sadece son zamanlardaki “kod yazmayı öğrenmek istiyorum” taleplerime, kendi yazdığı tutorial’ları okuyarak başlayabileceğimi söyleyerek, usta manevralarla kıvırıyor. Ama bir şeyleri kendi kendine kurcalayıp öğrenmek ve öğrendiklerinden bir şeyler üretmek çok zor şeymiş. Bunu yapabilenleri ne kadar takdir ettiğimi kelimelerle anlatmak zor.

Ve sonuç olarak bu 10 yıllık süreçten öğrendiğim en önemli şey, insanın kendini değerli hissedebilmesi için herhangi bir işte çalışıp para kazanmasına gerek olmadığıdır. Birazcık disiplin ve kararlılık ile bir şeylerin başına oturup, üretilen güzel şeylerden tüketip sonra da küçücük de olsa bir şeyler üretmek yetebiliyormuş.

**Şekil 2.**

Ve son olarak, demoscene kavramının bende nerden nereye geldiği:

1. Demoscene denen şeyden tamamen habersiz olduğum mutlu ve mesut yıllar
2. “Batsın bu demoscene, adını bile duymak istemiyorum” de-

diğim dönemler

3. “Bu demoscene de cok guzel bişeymiş, insan ne çok öğreniyor. Ama böyle boş boş durmamam lazım, bişeyler üretmeliyim” dediğim şimdiki zaman. Bu zamanın tatmini yüksek, ama hiç tasalanmadan tembel tembel günler geçirebildiğim dönemlere çok hafiften özlem duyduğum da bir gerçek :)

Scener eşi olmak zor işmiş :)

# Coder / Sanatçı Olmak Üzerine...

Cem (Spaztica) Gencer

Vakit bulup mola verdikçe web'de bulduğum güzel yazıları kahve eşliğinde okumaktan keyif alırım; ofis rutininden ve sıkıntısından kaçmak için de güzel bir yoldur. Ara ara ilgimi çeken siteler değişir, bazen LifeHacker, bazen ZenHabits... Son günlerde Y-Combinator girişimciler projesinin yöneticilerinden Paul Graham severek okuduğum birisi. Paul Graham'ı biraz araştırınca, uzun süredir kafamda belirginleşen bir konu hakkında bir kitap yazdığını gördüm: Hackers & Painters. Elbette alıp okumayı isterim, ama okumadan önce kişisel scene ve sanatsal eğitim geçmişimden yola çıkarak bir yazı yazmak istedim.

## Coder, Su Katılmamış bir Sanatçıdır

Programcılığın, programlamanın mühendislik olarak düşünüldüğü bakışın aksine, coder olan bir insanın mühendislikten çok bir sanatçı, bir yaratıcı olduğunu iddia edebilirim. Bunu biraz açalım... Coder her seferinde yeni bir şey yaratan, gereken verileri ve malzemesini topladıktan sonra derin bir konsantrasyon sürecine giren, kurguladığı kodları belleğinde yüzlerce kez farklı varyasyonlarla çalıştırabilen, gerektiğinde alt parçaların işleyişini takip etmek için bu süreci adım adım olarak da kafasında canlandırabilen kişidir. Metaforsal düşünme biçimi, ileri düzey programcılarda ve coder'larda çok kullanılır.

## Soyutlama, gerçek dünyadan kopmaktır

Soyutlama, son gelişen dillerde daha belirginleşen OOP anlayışının temelini oluşturur. OOP yapısında her parça, bütünden ayrı durabilen birer soyutlanmış nesnedir.

Sanatçılara baktığımızda önceleri kübizm, sürreelizm gibi akımlarla gerçeklikten ayrılan sanat anlayışı zamanla kavramsal, soyut, sosyal akımlara dönüştü. Hepsinin de temelinde belirli izlenimleri, düşünce ve duyguları kavramlar, metaforlar, görüntüler üzerinden soyutlayarak iletmek var. Üretim sürecinden önceki araştırma süreci aslında üretimin ne şekilde olacağını da kapsar. Her iki alanda da, coder da, sanatçı da, üretim evresine geldiklerinde normalden daha sancılı olabilen, çok daha yoğun konsantrasyon gerektiren sürece hazırlarlar kendilerini. Pek çok yazar da, kitaplarında dile getirdikleri evrenlerini sürekli akıllarında barındırırlar ve bu araştırma sürecinde bunu evirip çevirir, farklı gözlerle anlatmak istedikleri düşünceler örtüşünceye kadar değiştirirler.

## Sanat ve kod; mantığın iki farklı uzantısı...

Bir arkadaşla geçenlerde bu konularda sohbet ediyorduk; aslında sanatçı da, coder da temelde mantık üzerine kurulan farklı yapı taşları olarak algılanabilir. Kod yazma süreci, bu yapı taşlarını bir dil üzerinden ifade etmektir. Sanatın da dili soyutlamadır. Her ikisinin de dayandıkları temel, batı düşüncesinin temelini oluşturan bir pozitif bilim. Açıkçası daha algısal, mistik bir bakışla, sezgisel yaklaşım temel alınsaydı programlama nasıl bir noktaya giderdi, merak ediyorum. Fuzzy logic, kontrollü kaotik durumların kısmen hesap edilebilmesini sağlıyor. Peki tamamen organik bir sistemde bilgisayar teknolojileri nereye doğru gider, merak ediyorum açıkçası. Bilgisayar lojiği, 0 ve 1 üzerine kurulmuş bir sistemdir. Bu matematiksel ifade yöntemini -1, 0 ve 1 şekline çevirmek bile tüm bilgisayar teknolojisini değiştirebilecek kadar radikal. Bunun yapılabilmesi için kat edilmesi gereken çok yol var. Ben de konudan haylice uzaklaştım, farkındayım.

Bir süre önce yine web üzerinde bulduğum, ama artık yığın içerisinde bulamadığım bir yazıda bir coder'ın üretim sürecinde çok hassas bir durum olduğu ele alınıyordu. Düşünce akışı bir kere yüksek hıza çıktı mı, bağlantılar iyice soyut bir açıdan ele alınmaya başladı mı, o halde kod yazmak kadar keyifli birşey yoktur, yemek yemeyi bile unutur, saatlerce beyninizdeki kodları klavyeden diske akıtırsınız. Müzisyenlerin yoğunlaşıp improvizasyon yaptıklarında olduğu gibi, dışarıdan gelebilecek en ufak rahatsız edici etken -fotoğraf çekilmesi, yakında bir yerden ritimlerini bozacak bir düzensiz ritmik ses-, o düzlemden tepe takla düşmeye sebep olur. Kod yazarken komşu evden gelen çeşitli sesler, az ilerdeki okulun tenefüs zilinin çalması, eşinizin o anda içeri girerek akşam ne yemek yapacağını sorması sizi o anki düşünce akımını belleğinizde kaydedip -bazen buna fırsat bile bulamayabilirsiniz- dikkatinizi bu yeni etkene çevirmenize sebep olur. Ama etken ortadan kalktığında tekrar aynı konuma dönmeniz bazen çok hızlı, bazen de tüm kod işleyiş mekanizmasını yeniden canlandırmaya çalıştığınız için daha yavaş olabilir. Düzenli üretim yapan sanatçı ya da coder, bu geçişleri daha acısız atlatabilir, sürekli üretim yaptığı için her iki dünyada da aynı anda var olabilir. Sonucunda sanat da, kod yazmak da bir performans ürünüdür ve performans ne kadar sıklıkla tekrarlanırsa, o kadar kemikleşir. Düşüncenin sanatsal ya da kodsal dil aracılığıyla ifade edilmesi, farklı bir ülkenin diliyle konuşmayı öğrenen birinin ilk zamanlar yaptığı gibi kelime karşılıklarını hatırlayarak çevirmeye çalışmakla benzer bir gelişim eğirisine sahiptir. Pratik yapıldıkça bu karşılaştırma tabloları bırakılır ve o dil ile düşünülmeye başlanılır. Bence demo partileri de bu performansı, en çok dış etkende kesintisini bozmadan devam ettirebilme platformlarıdır. Bu etkinliklerdeki zaman limiti, sosyalleşme, sürekli fon müziği, video gösterimleri gibi etkenler, deneyimli coder için etken bile olmaz; klavyesinin başına oturdu mu, beynini tüm bu etkenlerden kapatmasını öğrenmiştir.

Kod ve sanat ilişkisi, benzerliği, aslında yapılan üretimlerin de devlet açısından benzer kriterde ele alınmasına yol açmış. Telif hakları kanunu, tekil, özel üretimlerde her ikisinin de Fikir ve Sanat Eserleri Kanununda tanımlanmış. Haliyle ticarileşme boyutuna girilmeden yapılan tekil işler, gerek sanat eseri, gerek yazılım uygulaması, devlet açısından da aynı ele alınıyor. Üretimlerin sonuçlarının ele alınma şekli de ayrı bir yazı konusu olabilir. Ama demoların birer sanat eseri oldukları, nitelikleri açısından ele alındıklarında daha da belirginleşiyor. Tekil üretilen bu demolarda coder'ın özgün hazırladığı kod parçaları dışında hiçbir şey tekrar etmez, her demo bunlar dışında sıfırdan hazırlanır. Demoscene'in telafuza gerek olmayan temel prensiplerinden biri özgün ve taze hazırlanan içerik. Son dönemlere kadar sanat eserleri de özgün olma konusunda iddiali bir pozisyona sahiptiler. Postmodernizmle gelişen yeniden kullanım anlayışı ile bir sanat eseri, diğerinin küllerinden / parçalarından / artıklarından doğabilmekte. 90'lı yıllarda kendini gösteren globalizm anlayışının dijital kültürdeki yansıması internetin gelişimiyle yeniden kullanım anlayışı da üretime yansıyor. Seksenli yıllarda özgün eserlerle başlayıp zamanla remix kasetlere, oradan da DJ kültürüne (Remix Culture) geçiş gibi bir evrim geçiren üretim anlayışı, daha eski bir anlayışta olan demo kültürünün varlığını sarsıyor. Haliyle demolarda içeriğin biricik olma durumu da çok önemsenmemeye, “sonsuz bir evren olan ağ”daki kaynaklar (Ghost In The Shell'e saygıyla...) sömürülmeye ve üretilenin kalitesi düşmeye başlıyor. Son senelerde yapılan üç boyutlu, efekt bazlı demoların hepsi, bu tür bir kalitesizleşmeden muzdarip. Partilerde üretimin kalitesel ölçümü gibi bir yöntem olsaydı, demoscene kültürü farklı bir yöne doğru gidebilirdi. Bazı gruplarda görülen art director’lük eğilimi demo gruplarının birer reklam ajansı gibi çalışması kalitesi yüksek ürünlerin oluşmasını sağlıyor.

Avrupa'da "Code is Art" düşüncesinin belirginleştiğini ve Ars Electronica gibi etkinliklerde sanat ve programlama arasındaki sınırın iyice eritildiğini görüyoruz. Internet'te sosyalleşmenin çekici olmaya, çeşitli API'lerle ve gelişen teknolojilerle melezleşen sitelerin de sosyal bağlara yönelmesi ile araştırmacı sanat teorisyeni-yazar Nicolas Bourriaud'ın İlişkisel Estetik kitabıyla ortaya attığı ilişkilerin oluşturduğu bir boyut düşüncesine paralel bir düşünce. Zaten OOP temelinde yer alan parent-child ilişkileri, inheritance gibi kavramlar bu yaklaşımın iyice yerleştiğini gösteriyor.

Üretim aşamaları olarak da, üretilenin ele alınışı ve bir süre sonra arşivlenmesi -müze ya da ftp server üzerinde- açısında kod yazmakla sanat eseri üretmek aynı yollardan geçiyor. Tek farkları, sanat eserinin daha kitlesel bir izlenme oranına sahip olması ve parasal dönüşüm değerine sahip olması.

# Türk Scene Tarihi - 2

## Uğur (Vigo) Özyılmazel

Öncelikle bazı düzeltmelerle yazıma başlamak istiyorum. Sevgili kardeşim Turbo/Bronx yazımı okuduktan sonra bana bazı hata yaptığım yerleri söyledi. Bununla da kalmadı, Türk Scene Tarihinin bazı gizli kalmış sırlarını da paylaştı.

Yazımın son paragrafında, bazı bilgisayar dükkanlarının da "grup" gibi çalıştığından bahsetmiştim. Özellikle "Uğur-Moda". Aynı bir scene grubu gibi, her sattıkları oyunun başına, "reklam yapmak" amacıyla dükkanın introsunu eklerdi. Aynı zamanda bu, "bakın, biz bu oyunu ilk getirttik, en yeni oyunlar bizde" nin de ifadesi idi. Keza o zamanlarda, oyunların başında duran intro'yu söküp, yerine kendi intro'nu koymak gerçekten çok büyük bir başarı olarak görülürdü.

Uğur-Moda'yı scene grubundan ayıran diğer bir özellik ise, oyunları parayla satın almasıydı. Yani grup gibi "swap" yapmaz, direk "HOTLINE" grubundan parayla oyunları satın alırdı. Keza başlarda "Metro Boys" ve "Fast Generation" da bu işi yapmıştı.

"Tacs from Turkey" aslında bir gruptan çok, Burak Kiper ( Beşiktaş'ta bulunan Erle Bilgisayar ) in kendi nick'i / adıydı. Tacs'ın introlarının büyük bir çoğunluğu DEF JAM grubunun source'larının değiştirilmesiyle yapılmıştı.

Zombie Boys'un üyerinden olan BODYS aslında 2 kişiden oluşan bir gruptu. Grup olarak Zombie Boys'a katılmışlardı. Keza "Music Masters" yada "Secret Members" nick'i ile Zombie Boys'a müzik yapan ikili; Sinan Bökesoy ( Halen müzisyen ve yapımcı olarak çalışmakta ) ve Erdem Taylan ( Garfield olarak da bilinir, Dreamworks'de efekt süpervizörü, Cars, Shrek, Ant-Z gibi 3d filmler ) dı.

Bu düzeltmeler ve ek bilgilerden sonra kaldığım yerden devam ediyorum.

Bilgisayarcılar ( aslında komik bir tabir, genelde oyun çektirdiğiniz yerdir! ) ile gruplar arasında ilginç bir bağ oluştu. İşin acı yanı, ana amacı yazılım satmak olan bu dükkanlar, o yıllarda tüm kaderini oyun dağıtan / getiren gruplara bağlamış gibiydi. Yani dükkanın ana amacı oyun satmak, yanında da eğer giderse joystick, disket, belki kartuş, disk drive gibi donanımlar satmaktı.

Biraz daha iyi durumda olanlar, ilaveten bilgisayar da satıyordu ama asıl konuları kaset / disket çekerek oyun satmaktı. Kimi dükkanlar grup seçerdi. Yani dükkanların sevdiği spreader'lar / grup'lar ve sevmedikleri vardı. Bu bakımdan dükkanlar bir şekilde gruplar tarafından kategorize edilip ayrılmıştı. Kimse kimsenin dükkanına gitmezdi. Taa ki bazı "ün" "şöhret" yapmak isteyen tipler çıkana kadar.

Grup mantığında çalışan diğer bir dükkanda TST idi. Teknoloji Sanayi ve Ticaret'in kısaltması olan TST scene tarihinde çok ciddi iz bırakmış dükkanlardandır. Grup Beyaz ve BS'den oluşuyordu. Amiga oyunlarının başına kendi yaptıkları TST introlarını koyarlardı. Keza TST donanım da üreten dükkanlardandı. En meşhur ürünleri, "competition pro" joystiğinden çakma "TST-JOYSTICK" dir. Özellikle "Kick-Off" oyuncuları bu joystiğin kıymetini iyi bilir ( Benim gibi! ) TST'den ileriki yazılarımda Türk BBS Scene'ini anlatırken de bahsedeceğim.


**Şekil 1.**

TST, oyunları LEVEL 4 grubundan ( Japan/LEVEL4 ) satın alıyordu. ( Alman grubu ) Keza Japan, aynı zamanda, o dönemlerde pek çok adı olan, benim hatırladığım adı MONITOR olan, daha sonra JASON olan ve son yıllarda SPAZTICA olan Cem Gencer'in de kontağı idi. Bu durum, Jason'ın Turbo ile tanışmasına vesile oldu.

Türk Scene Tarihinde mihenk taşı olan gruplardan Zombie Boys'un doğmasına büyük katkısı olan SORBİM de bizim meşhur dükkanlarımızdan. Move ve Ghost o yıllarda Sorbim'de çalışıyordu ve Turbo, Move'a, "SOURCE DISK"lerden götürdü. Defjam, Bamiga Sector1, Quartex'in source'ları. "SOURCE" ne diye merak edenler için yazıyorum; Assembler dilindeki kaynak kod. Assembler derleyicisi ile dosyaları açıp "compile" ederek "executable" ( yani çalıştırılabilir ) hale getirirsiniz. O yılların en moda kelimeleri "SOURCE" , "RUTİN" ( aslında bugün ki fonksiyon - function ) hayatımızın en önemli parçası olmuştu. "Abi, süper bi flood rutini yazdım!" gibi cümleler :) :) :)

Move'un Sorbim'den ayrılıp Mecidiyeköy'deki "Aytaç Bilgisayar" a girmesi, daha sonra bu dükkanın "Zombie Boys HeadQuarter" olmasına vesile oldu. İlk yazımda da belirttiğim gibi, bu tarihçe, ağırlıkla benim penceremden yansıtılan bir tarihçe oluyor. Bu bakımdan, eğer bu yazıyı okuyan eski scener'lar varsa bana

kendi anılarını yollamasını rica ediyorum. En azından daha objektif bir şeyler yazmış olurum.

O yıllarda en faal oyun dağıtan gruplar: Zombie Boys, Angels, Tacs, Metro Boys, Fast Generation, Turkish Mega Force gibi gruplardı. Oyunları getirmenin iki yolu vardı;

1. Crack yapan gruplardan parayla satın almak ( ki bunun için o gruplarla temasa geçmek inanılmaz zordu )
2. Swap yapmak, iyi kontaklar kurmak.

2.yol haysiyet açısından daha “kool” bir davranıştı. Eğer “baba” bir ekipsen iyi kontakların olur ve sana her zaman “yeni” stuff'lar gelir. Tabi işin zor kısmı bu kontakları kaybetmemek için seninde onlara iyi stuff'lar yollaman gerekir. Sonuçta swap'in mantığında, sana gelen stuff'ı diğer kontağa yollamak yatar. Yani Ali'den geleni Veli'ye, Veli'den geleni Ahmet'e yolla... Önemli olan stuff'ların çakışmamasını sağlamak. Çünki siz bir mektup yolladığınızda bu mektubun karşı tarafa ulaşması en az 1 hafta, karşıdaki kişinin size yollayacağı disket ( ya da disketler ) i hazırlaması da 1 hafta, size postanın ulaşması en az 1.5 - 2 hafta, yani sonuçta 1 ay gibi bir süre söz konusu.

Avrupa'da yaşayan gruplar bize göre kendi aralarında çok daha hızlı mektuplaşabiliyorlardı. Bu bakımdan sizin onlara gönderdiğiniz şey gerçekten hem hızlı olmalıydı hem de kaliteli. Bir süre sonra elinizde gönderecek bir şey kalmayınca ne olacak? İşte bu durumda artık sizin, kendi grubunuza ait bir ürün çıkarmanız ve tüm kontaklarınıza bu ürünü ya da ürünleri yollamanız gerekecekti.

Artık kontakları mutlu etmek ve onların gözünde “iyi” olmak için Türk Gruplarının da bir şeyler üretmeye başlaması gerekmekteydi. Commodore 64 grupları içinde gerçekten kendi introlarını yazan, dağıtan ekipler vardı. Büyük bir kısmı da ya başka grupların ( yabancı ) intro packerlarını kullanıp kendi logolarını koymaya çalışır, ya da direk olduğu gibi intro'yu çalıp renklerini, becerebilirse müziğini ve grafiğini değiştirerek kendi yapmış gibi dağıtırdı.

Bu gruplardan hatırladığım “COBRA BOYS” vardı. “MOSKWA TV/Beastie Boys” tarafından ( bir rivayete göre moskwa tv'de büyük ripper'lardanmış ) yapılan bir intro'yu aynen değiştirmiş ve “Cobra Boys” intro'su yapmıştı.

“Dükkan-Grup” lara en iyi diğer bir örnek de “Flamingo Bilgisayar” dır. İstanbul / Şişli'de dükkanı bulunan Flamingo, diğer “bilgisayarcı”lara nazaran intro işine çok önem verirdi. Çünkü grubun introları, daha sonra Zombie Boys'un C64 lideri olacak olan Mephisto/Comrade tarafından özenle yapılmış intro'lardı. Benim gözümle gördüğüm yaklaşık 4 disket ( arkalı - önlü ) Flamingo Intro Packer'ları vardı.

Genel anlamda 1989-1994 yıllarına dönüp baktığımda, irili ufaklı pek çok grubun var olduğunu, bunların çok büyük bir çoğunluğunun sadece ve sadece “oyun dağıtan grup” statüsünde olduğunu görüyorum. İlk hedef her zaman ünlü olmak, isim yapmak, en azından dükkanlar tarafından tanınan bir ekip olmaktı. O dönemlerde, “korsan” tabiri neredeyse hiç bilinmediği gibi, neredeyse, tüm bu dükkanların gelir kaynakları, sattıkları “korsan” oyunlar sayesinde kazanılmaktaydı. Bu durum, oyun dağıtan grupların ister istemez birbirleriyle ciddi rekabete girmesine yol açıyordu.

Türkiye'de “scene” kelimesi tam olarak telaffuz edilmese de, yurt dışında oyun kırmak, kırılan oyuna intro eklenmesi gibi işlemler biraz daha ileri seviyeye ulaşmış, artık yavaş yavaş intro'lardan oluşan “intro-collection” tarzında “release” ler çıkmaya başlamıştı. Internet'in ve modem'in henüz icad edilmediği bir dünyada iletişim “posta” yoluyla yapılmakta, insanların birbirini bulabilmesi için de “disk-mag” adı verilen “disket dergi”leri yavaş yavaş yayılmaya başlamaktaydı...

Hiçbir kaynağın bulunamadığı Türkiye'de, hasbel kader ( yani şans eseri ) disk-mag bulan ya da görenler, eğer birazda İngilizce'leri varsa, hemen dergideki “contacts” ya da “advertisements” bölümlerine dalıp, gördükleri tüm adreslere mektup göndermeye başladı. Tabi bazı ekipler, bunu çok önceden çözmüştü ve zaten yurt dışı ile çok sıkı irtibat içindeydi.

Aklıma ilk gelen efsane isim “Tolga Ayvaz”dır. Bir rivayete göre Tolga Ayvaz, bir gün Kadıköy/Yazıcıoğlu iş hanındaki “Uygar Elektronik” e ( keza bu dükkan aynı isimle halen orada bulunmakta ama eski günlerden hiç kimse bulunmamaktadır. ) gider, ( 3.kat ) “Sinan Kömürlü”nün ofisine girip, yanından getirdiği bir çuval disketi yere dökerken “abi yeni oyunlar geldi” der...

Tabi tüm bu “grup” tadındaki ekiplerin genelde tek ortak yönü sadece “import” a yönelik işler yapmasıdır. Yani genel sorunumuz olan “üretmeden tüket” mantığı bence taaaaaa o yıllara dayanır. Bu tarz ekipler daha “business” takılırken, diğer küçük çaplı ekipler, ufak ufak intro code'lamaya başlamış, hatta tam olarak bunu neden yaptığını bile bilmeden, kendini “dükkan” gibi görüp, oyunların başına reklamlarını yapmaya başlamıştır.

“Swap” yapmak ( yani posta yoluyla değiş-tokuş ) zor ve zahmetli bir iştir. Onlarca disketi kopyalamak, kimi özel kontaklar için uzun mektuplar yazmak, “stamp cheating” yapmak, mektupları zamanında postalamak, bunları takip etmek filan çok zaman alan ciddi bir iştir. Keza, dünya ile tek bağlantı bu metot olunca, işin önemi bir kez daha kendini hissettirir.

1990-1991 yıllarına baktığımda, bilgisayar dergilerinin konunun yayılmasına ne kadar katkı sağladığını görüyorum. Efsane dergi “Commodore”, daha sonra çıkan “Amiga Dünyası” gibi dergiler, belki de “Commodore” markasını Türkiye'ye getiren “Teleteknik”in yapması gerekeni yapıyor, onların misyonunu üstleniyordu. ( Keza “Commodore” dergisi de zaten Teletekniğindi )

Benim “scene” ve “scene grubu” kelimesinden anladığım, prodüktivite ağırlıklı, bol bol ürün çıkan / çıkarılan, içerisinde departmanların olduğu gruplardan oluşan bir oluşumdur. İçerisinde programcı ( coder ) bulunduran, müzisyen ( composer ), grafiker

( gfxer ), swaper, trader, spreader bulunan ekip tam bir “scene” grubudur. Yüksek ihtimalle bu tarz gruplar, en azından her seferinde “0-days old” ürün sahibi olabilmek için kesin bir “disk-mag” de çıkarmışlardır.

Commodore 64 ile birlikte artık Amiga da iyiden iyiye Türkiye pazarına girmiş, çok ciddi bir şekilde kullanıcı sayısı da artmaya başlamıştır. Yeni platform, direkt yeni oluşumlara yol açmış, aynen C64'de olduğu gibi Amiga içinde “grup” kavramı “release” , “import” kavramları ortaya çıkmıştır. C64 gruplarının bir kısmı hemen olaya adapte olabilmiş, hemen “Amiga” da da grup kurmaya, oyun dağıtmaya başlamıştır.

Zaman içindeki tüm bu gelişim süreçleri, her zaman olduğunu gibi, yine yurt-dışı aktivitelerine paralel bir şekilde ilerlemek durumunda kalmıştır. Yani başka bir değişle el oğlu ne yaparsa, Türk gençleri de onları örnek alıp taklit etmek zorunda kalmıştır. Bunun bence en büyük sebebi, Türkçe kaynak eksikliğindendir. Her zaman Almanya ve Alman grupları, bence, hep bir adım ötede olmuşlar ve olmaya da devam etmişlerdir.

Oyun kırmak, oyunları sonuz haklı yapmak, kaset versiyon oyunları disk versiyona çevirmek, oyunun başına en güzel intro'yu koymak, hatta oyundan ziyade, yapılan intro'nun kalitesi ve güzelliği ile övünmek yurt dışında ön plana çıkmıştır. Intro'lar yardımıyla gruplar, programlama seviyelerini, görsel ve işitsel yeteneklerini ispatlamak istemişlerdir. Bu rekabet ve hırs, bugün “demo-scene” dediğimiz şeyin doğmasını sağlamıştır.

Yani oyun kıranlar, bir şekilde daha sanatsal bir yola doğru yönelmişlerdir. Bu noktada artık “grup” kisvesi altındaki ekipler, daha farklı şeyler üretmek zorunda kalmışlardır. “Oyunu en hızlı biz import ettik” demek yerine, “yeni intromuzu gördünüz mü?” gibi cümleler yavaş yavaş Türk gruplarını da etkisi altına almaya başlamıştır.

“Release yapan” gruplar kimlerdi ve release etmişlerdi diye düşündüğümde; olayları ikiye ayırıyorum, sadece intro release edenler ve gerçekten uğraşıp “demo” release edenler diye. İlklerin bilgisayarı olan “C64” de ilk “mega-demo” denemeleri The Metro Boys'dan geldi desem sanırım hata yapmış olmam. 3 part'dan oluşan demonun bence en ilginç bölümü, IRQ-LOADER'ı olmasıydı. 2.part ekranda oynarken arka plandan 3.part'ın yüklenmesi ve “SPACE” tuşuna basarak direkt 3.part'a geçiş... Genelde bu tarz demolar ( ki bunlar ağırlıkla arka arkaya introların pack edilmesi ile olurdu ) space ile geçtikten sonra diğer bölümü yüklerdi... The Metro Boys'un bence 2 eksiği vardı, grafiker ve müzisyen...

Pek çok intro vardı piyasada fakat bir konsept içinde, başı - sonu olan, adı olan “demo”lar henüz hayatımıza girmemişti. Diğer bir dezavantaj da, zaten bu demoları kim izleyecekti ve neden izleyecekti??? Nereden bulacaktı??? En azından intro'ları oyunun başında görmek daha kolaydı. Hem diğer gruplarının birbirini bulması ya da birbiriyle yarışması açısından, hem de sokaktaki vatandaşın kolayca görmesi açısından.

En azından herkes oyunları bilgisayarcıdan satın alıyordu ve aynen bugünkü “reklamlar” mantığında, oyun yüklendiği an ekrana intro çıkıyordu. Anlayan anlamayan en azından 5-10 saniye intro'yu görüyor ve içgüdüsel olarak “space” tuşuna basıp intro'yu geçiyordu. İşte bu noktada yapılacak en akıllı şey, intro'yu TÜRKÇE olarak hazırlamaktı. Heyecanla oyunu oynamak isteyen kişi, ekranda, biranda çıkan TÜRKÇE bir şey görünce, içgüdüsel olarak “hmmmm bu ne yahuuu?” diyip merak ediyor ve bir süre intro'ya bakıyor ve hatta kayan yazıları dahi okuyordu.

Bu bakımdan “demo” nun mecrası, sadece yurt-dışındaki scene'di. Bu bakımdan, uluslararası standartlara uymak gerekiyor, İngilizce demo hazırlamak gerekiyordu. Keza yapılacak üretimler boşa gitmeyecek, geriye “stuff” olarak dönecekti. Kimisi yeni demolar, tool'lar kimileri de yeni oyunlar olacaktı.


**Şekil 2.**

Bir konsept içinde çıkan, multi-part olan ilk C64 mega-demosu, Zombie Boys'un yaptığı “ABAZOMANIA” adlı üründür. Yanılmıyorsam tarih 1990'dır. İncelendiğinde, aslında 5-6 intro'nun arayüklemesinden oluşan bir koleksiyon şeklindedir. Sadece 2 part bu demo için özel hazırlanmıştır. Müziklerin hepsi “rip” ( o an grupta C64 müzisyeni yoktur ), pek çok Zombie Boys logosu, karakter setlerinden yapılmıştır. Demo'nun en meşhur part'ı son part'dır. “Real Ghost-Busters” oyunundan sökülmüş gece-şehir silüeti, Amiga grafikeri Turbo/Zombie Boys'un çizdiği özel logo ve smily sprite'ları C64 için ilklerden sayılır.

Zombie Boys'un C64 bölümü, bu mega-demo ile çok ciddi başarı sağlamıştır. İşin komik yanı, çok da büyük bir yapım değildir. Grubun kontak sayısı biranda 100'ü geçmiş, “stuff” yağmuru başlamıştır.

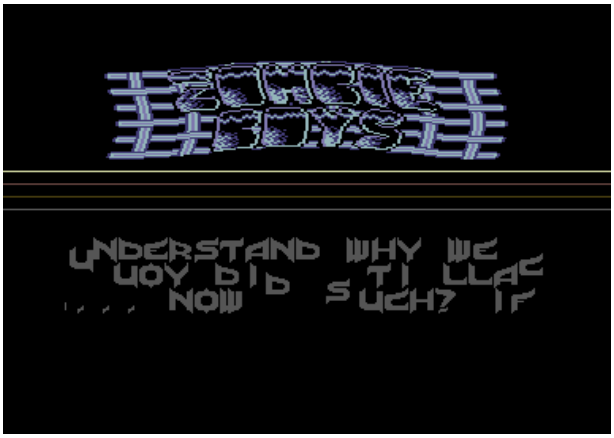
Kendi introsunu yazabilen pek çok diğer Türk grubu ( pek çok derken sayı maksimum 4-5 dir ) nedense mega-demo olayına fazla rağbet etmemiştir. The Joker Crew, Cobra Boys, TNF-77, The Metro Boys, Fast Generation, Light Force gibi gruplar sadece intro yapmakla yetinmiştir. ( Keza Echo Crew , Pet Shop Boys, The Rough Boys vs... )


**Şekil 3.**

Zombie Boys'a Mad - Master ve Prince üçlüsünün katılmasıyla, belki de C64 demo tarihinin ( tüm dünya çapında ) en önemli demolarından biri olan “Midnite Movie” mega-demo'su çıkmıştır. Gerçekten de tam bir uyum / tema / konsept içinde, konusu olan, pek çok yeni hatta hiç yapılmamış efekti barındıran bu demo, dünya çapında ses getirmiş, Türk grupları içinde bir ilki başararak yabancı disk dergilerindeki “mega-demo charts” lara girmiş, hatta bazı dergilerde 1 numaraya kadar da çıkmıştır.

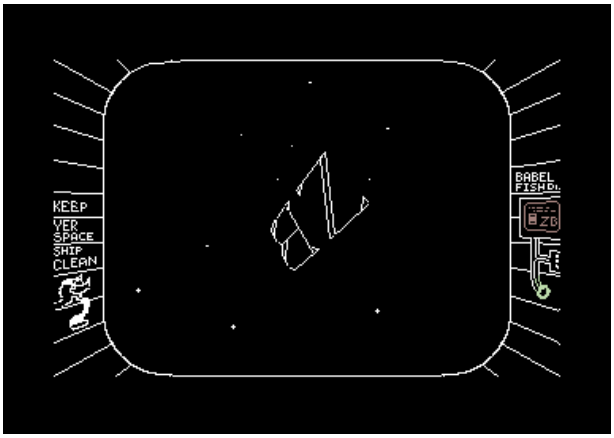

**Şekil 4.**


**Şekil 5.**




**Şekil 6.**

Bu muhteşem demo, gruba pek uğurlu gelmemiştir ne yazık ki... İlk kez bir Türk grubunun, yabancı grupların henüz yapamadığı efektleri yapması, dışarıda pek çok kişiyi kıskançlık krizine sokmuştur. Bir şekilde Avrupa'lı scener'lar, bu demonun “çalıntı rutinler” den oluştuğunu düşünmüş, hatta pek çoğu da demoyu didik didik “debug” edip bir şeyler bulmaya çalışmış ama hiç bir falso bulamamışlardır. ( Vektör part'daki çizgi çizdirme fonksiyonu yanılmıyorsam Uridium adlı bir C64 oyunundan esinlenilmiştir. )




**Şekil 7.**

C64 platform'undaki diğer demo çıkaran gruplar ise; Clique , Accuracy, Aesrude, Glance gruplarıdır. İşin ilginç yanı Commodore Scener Database ( CSDB ) sitesine baktığımda, Türkiye'den 35 grup görüyorum. Hiç bilmediğim gruplar da var... “The Phaze Company” adlı grup 1989 yılında “one-file-demo” olarak 2 adet intro ( üst üste pack edilmiş ) dan oluşan bir demo çıkartmış. Tabi bu 35 grup içinde küçük küçük introları olan pek çok grup mevcut. Benim ilgilendiğim kısım sadece DEMO / MEGA-DEMO şeklinde olanlar.

Zombie Boys grubunun adını BRONX'a çevirmesi ile yepyeni bir oluşum başladı. Commodore 64 ve Amiga platformunda pek çok şey üretildi. Grup büyüdü. Diğer yabancı ülkelerden ve İstanbul dışındaki şehirlerde de katılımlarla sayısı 100'ün üzerine çıkan üyeleriyle BRONX dünya çapında tanınan bir grup oldu.


**Şekil 8.**

Grup içinde yaşanan sorunlardan dolayı İzmir ve Ankara bölümleri gruptan ayrılarak Clique grubunu kurdular. Bu grup C64'de bir demo release etti : Freestyle ( 1991 ) (Editör: Clique asıl başarısını Script adlı Diskmag'leri ile elde edecekti).

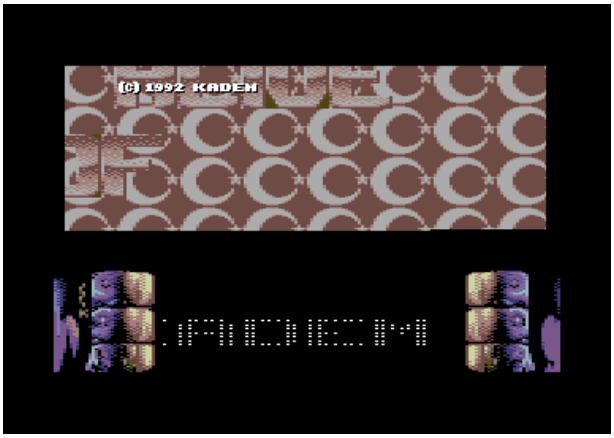

**Şekil 9.**


**Şekil 10.**

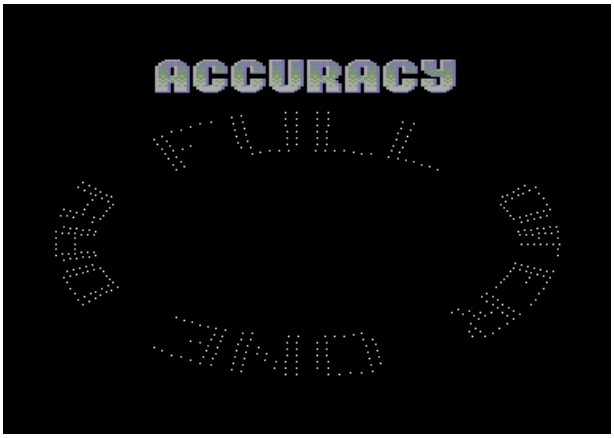
Clique dışında 3 tane demo release eden diğer bir C64 grubu da Accuracy'dir;

1. Color of Code ( 1992 ) (Editör: Beni en gazlayan demolardan biridir)
2. One Year Accuracy ( 1990 )
3. Unstopable ( 1990 )
4. Flabbergarsted ( 1989 ??? ) ( Tarih biraz muamma )
5. Thunder ( ??? )

Keza, hafızam beni yanıltmıyorsa bu grup aslen Avusturya menşeli bir gruptu. “Frib/Acy” adlı bir swaper vardı. Yanılıyor olabilirim.




**Şekil 11.**




**Şekil 12.**

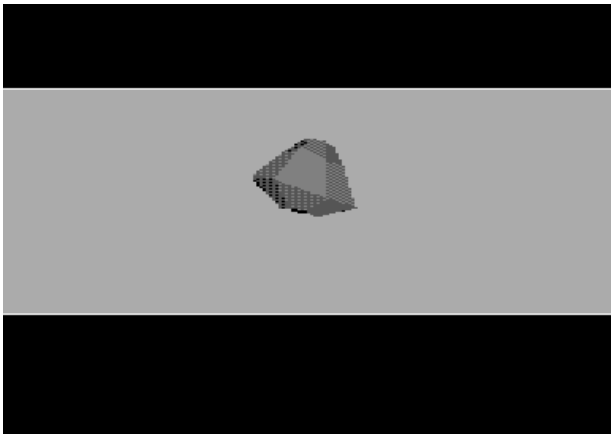

**Şekil 13.**


**Şekil 14.**

2003 yılında inanılmaz birşey oldu ( Gördüğünüz gibi 1990'lardan 1991'lerden direk 2003 atladık ) Aesrude adlı bir Türk C64 grubu, “Water” adlı demosuyla Forever partisinde 1.oldu. Yaklaşık 13-14 sene rötarla scene'e giren, girdiği gibi ortalığı dağıtan Nightlord, tek başına ( code , gfx, sound ) bir fabrika gibi tozu dumana kattı.(Editör: Estafurullah yok öyle birşey... Sanırım Vigo'ya bu satırlar için bir yemek ısmarlamalıyım )

**Şekil 15.**

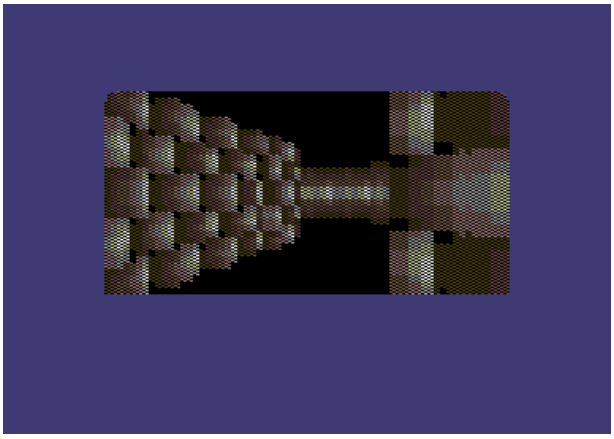
Zaman çizelgesinde baktığımızda ( Commodore 64 için ) 1989 , 1990 , 1991 , 1992 yılları dolu dolu yaşanmış, sonra PAUSE olmuş, tam 11 yıl beklenmiş gibi bir durum söz konusu. ( Tekrar ediyorum, DEMO release eden grupları kastediyorum sadece )



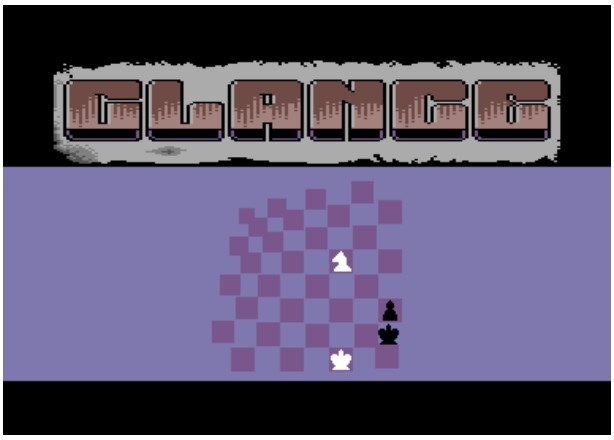
**Şekil 16.**

2 senelik bir PAUSE sürecinden sonra yıl 2005 olur ve daha büyük bir bomba patlar. Gelmiş geçmiş en iyi C64 demolarından biri olmaya aday olan “Living”, Glance adlı grup etiketiyle çıkar. Tamamen Türk scener'lardan oluşan Glance, 7D5 Demoscene Party'de Demo-Compo'yu kazanır.

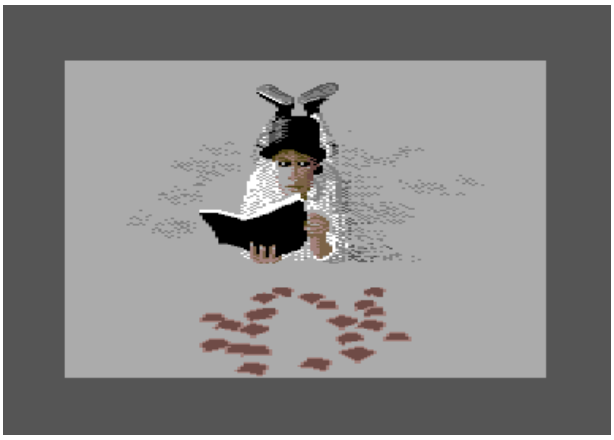

**Şekil 17.**



**Şekil 18.**




**Şekil 19.**



**Şekil 20.**

1989 - 2008 arasına baktığımda ( 19 yıl ) Commodore 64 platform'unda DEMO release etmiş hepi-topu 6 grup bulunmaktadır. Bunlar;

1. Zombie Boys
2. Bronx
3. Accuracy
4. Clique
5. Aesrude
6. Glance

19 yıla 3 aşağı 5 yukarı 10-15 adet DEMO / MEGA-DEMO serpiştirilmiştir. Yurt dışı ile karşılaştırılınca durum çok ciddi şekilde vahim. Tabii “dentro” ya da “intro” , “intro collection” , “music disc” , “graphics-collection” gibi ufak çaplı ürünler olarak bakınca tonlarca şey bulmak mümkün... (Editör: Diskmag kategorisinde Türkiyeden çıkmış önemli ürünler var fakat diskmag konusunu belki de başlı başına ayrı bir yazıda irdelemek lazım)

Önümüzdeki sayıda bu 19 yıllık süreçte, Amiga platformunda Türkiye cephesinde neler oldu? Hangi grup / gruplar çıktı, neler üretti bunları yazmaya çalışacağım. Daha sonraki sayıdaysa bildiğim kadarıyla PC ve diğer platformlardaki gruplara ve çıkan ürünlere bakacağız yine bu 19 yıllık süreç içinde...

# Plazma Künye

## Plazma

Amatör Bilgisayar Kültürü

http://www.plazma-dergi.org

Sayi 5 : 15 Nisan 2008

## Ekip

### Yazarlar:

- Ahmet Zeki Eymür
- Arda Karaduman (CoZe)
- Arda Ö. Erdikmen (Ref/Crescent)
- Arnold Cistai (Jailbird/Booze Design)
- Bilgem Çakır (Nightlord/Glance^Aesrude)
- Cem Gencer (Spaztica)
- Emir Akaydın (Skate/Glance^Clash)
- Emir Diril (Drey/Demodojo)
- Emirhan Bayyurt (Ragnor/Clash)
- Gökhan Sönmez (LW3D/Bronx)
- Kürşad Karamahmutoğlu (Hydrogen/Glance)
- Özgür Yıldırım (Lord Henry Wotton II)
- Seval Çakır (Irian/Aesrude)
- Şemseddin Moldibi (Endo/Glance)
- Türker Gürevin (Alcofribas)
- Uğur Özyılmazel (Vigo/Bronx)

### Kapak:

- Kürşad Karamahmutoğlu (Hydrogen/ Glance)

### Yardımcı Editörler:

- Haber Editörü: Emir Akaydın (Skate/Glance^Clash)
- Kürşad Karamahmutoğlu (Hydrogen/Glance)

### Editör:

Bilgem Çakır (Nightlord/Glance^Aesrude)

## İletişim:

Plazma'da yazar olmak istiyorsanız veya dergi ile ilgili görüşlerinizi bizimle paylaşmak isterseniz, aşağıdaki adres aracılığıyla editörlerle temasa geçebilirsiniz.

nightlord (at) nightnetwork (nokta) org