

004.64

6975

1996

T.C.
MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRONİK VE BİLGİSAYAR EĞİTİMİ
YÜKSEK LİSANS TEZİ

UZAKTAN YÖNETİLEBİLEN MODEM
BİLGİ İŞLEM YAZILIMI

BARBAROS YILDIRAY GÜNAY

TEZ DANIŞMANI
PROF.DR. MACİT GÜNEŞ



Marmara Üniversitesi
Kütüphane ve Dokümantasyon Daire Başkanlığı



T03916

İSTANBUL, 1996

İÇİNDEKİLER

ÖNSÖZ		4
1 - Özet		6
2 - Giriş		7
2.1	Çalışmanın eğitim amaçlı analizi	7
2.2	PC Donanım Yapısı	8
2.2.1	Elektronik Tümlşik Devrelerin Evrimi	8
2.2.2	PC-Bilgisayarların Evrimi	9
2.2.3	IBM Uyumlu Bilgisayarlar (Personal Computer Clones)	11
2.2.4	IBM'le yol ayrımı ve Endüstri Standardı bilgisayarlar	12
2.2.5	Kişisel Bilgisayarların Bileşenleri	13
2.2.6	Ana Donanım Bileşenleri (Main Hardware Components)	15
2.2.7	BUS (YOL.) Anlamı, gereği ve farklı "yol" standartları	16
2.2.8	CPU	21
2.2.9	Kesmeler	29
2.2.10	Çevre Kontrol Üniteleri	30
2.3	PC Yazılım Yapısı	35
2.3.1	Komut Seti	35
2.3.2	BIOS	38
2.3.3	İşletim Sistemi	40
2.3.4	Kesme Yönetimi	40
2.3.5	Resident Programlar	41
3 - Programın Çalışma Şekli		42
4 - Projede İzlenen Yol		43
4.1	Yakın PC'den uzak PC'ye yakın PC'de basılan tuş bilgilerinin gönderilmesi	44
4.2	Uzak PC'den yakın PC'ye ekran değişimlerinin gönderilmesi	45
4.3	Uzak PC'nin kendisine gelen tuş bilgilerini algılaması	46
4.4	Yakın PC'nin kendisine gelen ekran bilgilerini algılaması	47
4.5	İletişimin modem üzerinden kurulması	48
4.6	Proje parçalarının birleştirilmesi	49
4.7	İleri çalışma önerileri	49
5 - Sonuç		50
6 - Kaynaklar		50

7 - Ekler

Ek 1	Yöneten bilgisayarda basılan tuşları gönderen program	51
Ek 2	Yöneten bilgisayara gelen ekran bilgilerini algılayan program	55
Ek 3	Yönetilen bilgisayarın ekran değişimlerini gönderen program	59
Ek 4	Yönetilen bilgisayara gelen tuş bilgilerini algılayan program	72



ÖNSÖZ

Haberleşme bir elektronik mühendisliği konusu olarak başlamış, son yirmi yılda bilgisayar teknolojisinde yaşanan korkunç hızlı gelişmeler sonucu bilgisayar temelli aletlerin birbiriyle bağlanması konusuna dönüşmüştür. Başlangıçta sadece donanım önemliyken, son yıllarda yazılımın haberleşmedeki önemi artmış ve minimum donanımla haberleşme sistemleri gerçekleştirme çözümleri aranmaya başlanmıştır.

Bilgisayar sistemleri boyut olarak küçüldükçe ve client/server mimarisinin önemi artmıştır. Artık büyük şirketler mainframe ve buna bağlı terminaller kullanmak yerine, büyük kapasiteli orta boy bir bilgisayar ve uzak noktalarda buna istendiği an bilgi gönderip alan PC tabanlı akıllı bir bilgisayarlar kullanmayı tercih etmektedirler. Bu yapıda bütün yükü üstlenen aradaki haberleşme protokolü olmaktadır.

Haberleşmenin hatasız gerçekleşmesi için çeşitli yapılar ve protokoller önerilmektedir. Bu protokollerin içinde hata tanıma veya hata düzeltme kodları da bulunmaktadır. Son yıllarda ise hata düzeltmenin yanında bilgi sıkıştırma da önem kazanmıştır. Bilgiyi akıllı bir şekilde sıkıştıran yazılımlar geliştirilmiştir. Öyle ki bir resmi değerlendiren bilgisayar, resmin her noktasını aktarmak yerine resmin siyah kıvrıkcık saçlı, kahverengi gözlü bir adama ait olduğunu aktarabilmektedir.

Eskiden büyük kapasiteye ulaşan bilgilerin boyu sıkıştırma sonucu küçüldükçe dünya çapındaki ağlarda bu bilgilerin aktarılması kolaylaşmakta ve bu tip ağlar gelişmektedir. Çağımız haberleşme ve bilgi çağı haline gelmiştir. Herkes kendini bilgiye aç hissetmekte ve bilgiye ulaşmak için büyük paralar ödemektedirler.

Yazılım projeleri haberleşme sektöründe yoğunlaştığından bu sektöre büyük yatırımlar yapılmaktadır. Dünyadaki üst düzey mühendislerin çoğu bu alanda veya yan alanlarında çalışmaktadır.

Bu unsurları gözönünde bulunduran çok uluslu firmalar yazılım üretmek için ucuz üretim maliyeti olan ülkeler aramaktadırlar. Şimdiye kadar Hindistan, İsrail gibi ülkeler bu konuda ön plandayken artık Türkiye de bu tip çok uluslu firmalara üretim yapan ülkelerden biri olmuştur. Dünyada tanınan bir çok ünlü haberleşme firması büyük projelerinin bir kısmını gerek Türkiye'de gerek Türkiye dışında Türk mühendislerle birlikte gerçekleştirmektedirler.

Bilgisayarlar arası haberleşmenin önemini düşünerek oluşturduğum yazılım birbirinden uzak noktalarda yer alan iki bilgisayarı birbirine bağlamak için kullanılan ve en yaygın örneği CarbonCopy olan bir yazılımdır. Bağlantı kurulduktan sonra uzak noktadaki bilgisayar yakın noktadaki bilgisayarın tuş takımı yardımıyla kullanılmaya başlar. Böylece bir kişinin uzak noktaya gidip yapacağı bir işi kendi iş yerinden yapması sağlanır.

Bu tip bir program günümüzde çok uluslu veya ülke çapında bir satış ağına sahip destek firmaları tarafından kullanılmaktadır. Böylece firma elemanları zamanlarının büyük bölümünü yolculukta geçirmekten kurtulmakta ve firmanın yolculuk giderleri önemli miktarda düşmektedir. Aynı zamanda arızaya müdahale zamanı çok kısaldığından servis kalitesinde de büyük bir yükselme olmaktadır.



1 - Özet

Birbirinden uzak noktalarda bulunan iki bilgisayarı birbirine bağlamak için iki yaygın yöntemden biri kullanılır. Bu yöntemlerin en ucuz ilk maliyetli olup, en yaygın olarak kullanılanı uzak bilgisayarın diskini yakın bilgisayarın herhangi bir sürücüsü olarak göstermektir. Bu yöntem dosya transferi yapılacak ise ucuza gelmekte fakat uzak bilgisayarda bir program çalıştırılacaksa çok yavaş ve pahalı olmaktadır. Bu yöntemi kullanan programlar son DOS versiyonlarının içinde ücretsiz olarak bulunmaktadır.

Aynı zamanda internet üzerinde çalışabilen NFS yazılımı yine bu tip bir yazılımdır. Sadece dosya transferi amacıyla internet üzerinde çalışabilen FTP(file transfer protocol) adlı bir diğer yazılım da kullanılabilir. Her iki yazılım da TCP/IP katmanının üzerine yerleşerek bir dosya ve directory transfer protokolü geliştirmişlerdir.

Benim kullandığım ve daha pahalı yazılım maliyeti olan yöntemde ise uzak noktadan yakın noktaya sadece ekran transferi yapılmakta, yakın noktadan uzak noktaya ise basılan tuş bilgilerini gönderilmektedir.



2 - GİRİŞ

2.1 Çalışmanın eğitim amaçlı analizi

Bu çalışmada incelenen konu oldukça geniş bir konudur. Yazılım, çalıştığı ortam nedeniyle çok iyi bir alt düzey PC bilgisi gerektirmekte, bir yandan da çok fazla bileşeni bir araya getirdiğinden iyi bir tasarım tecrübesi gerektirmektedir. Bu çalışmanın giriş bölümünü PC donanımını ve yazılım sistemini tanıtmaya ayırdım. Bu bilgiler bir çok kitapta dağınık şekilde bulunabilmektedir. 3. (Programın Çalışma Şekli) ve 4. (Projede İzlenen Yol) bölümde ise programın bileşenleri tanıtılmakta ve bu bileşenlerin hangi mantıkla bir araya getirildiği incelenmektedir.

Bu tip bir yaklaşım günümüzde component tabanlı programcılıkta çok sık kullanılmaktadır. Component tabanlı yazılım geliştirmek isteyen yazılımcı, yapacağı işin temel parçalarını, daha önce aynı parçaları geliştirmiş olan bir başka firmadan satın alır. Kendisi sadece küçük ekleri yapar. Günümüzde Visual Basic tabanlı Windows programcıları bu tip componentleri kullanmaktadırlar.

Benim geliştirdiğim yazılım da bu anlamda iki ayrı component'ten oluşmaktadır. Örneğin bu parçalardan biri klavyeden bilgi okurken, diğer parça seri porttan gelen ekran bilgilerini ekrana yöneltmektedir. İstendiği an bu parçalardan biri bütünden ayrılarak bir başka programın parçası olarak kullanılabilir. Bir başkasının programını kullanmak bir programcının eğitiminin önemli parçalarından biridir.

Tasarımdaki, donanım cihazlarını kullanma şekli de eğitimde kullanılabilir bir uygulamadır. Yazılımda tüm kesmeler yakalanmakta ve bazıları üzerinde ilginç yönlendirmeler yapılmaktadır. Tuş takımından gelen tuş bilgilerini seri kanala yönlendirmek buna bir örnektir. Böylece o bilgisayara sıkı sıkıya bağlı kaynaklardan gelen bilgilerin, mutlaka o bilgisayar içinde kullanılmak zorunda olmadığı görülebilir.

2.2 PC Donanım Yapısı

2.2.1 Elektronik Tümüleşik Devrelerin Evrimi

Yarı iletken üretim teknolojilerinde başlangıçtan (1950'li yıllar) itibaren elde edilen gelişmeler sonucu, bir çok elektronik işlevi yapabilen, ve giderek daha çok sayıda yarı iletken devre elemanının (Diyot ve Transistör) bir paket içinde üretilebilmesi gerçekleşti. 1960'lı yıllarda piyasaya çıkan bu paketleme genelde Tümüleşik devre veya Entegre devre veya kısaca entegre (çip) olarak adlandırılır.

1970'li yıllardan başlayarak Amerikada yarı iletken üreten bir çok firma bu tümleştirme teknolojisini kullanarak, aritmetik ve mantik işlemler yapabilen entegreler üretti (ALU- Arithmetic and Logic Unit). Bu tümleşik devreler daha sonra, Merkezi işlem birimi (Central Processing Unit) veya kısaca CPU olarak isimlendirildi.

Günümüzde dahi bir çok firma kendi özgün tasarımlarında CPU üretmekle birlikte biz daha çok 80x86 serisini üreten Intel (Integrated Electronics) ve 680x0 serisini üreten Motorola firmasını tanıyoruz.

Intel 1971 yılında 4004 çipini üretti. Yaklaşık 2300 transistör içeren bu çip sadece 4 bit'lik Nibble'lar (küçük parçacık veya lokmacık) üzerinde işlem yapabildiği için yaygın kullanım alanı oluşturamadı.

Intel 1972 de 3500 civarında transistör içeren 8008, 1974 te 6000 transistörüü 8080, 1975 te 6500 transistör içeren 8085 çiplerini üretti. İsimlendirilmelerinden anlaşılacağı gibi bu çipler 8 bitlik veriler üzerinde işlem yapıyorlardı.

70'li yıllarda yine üretim teknolojilerinde sağlanan gelişmelerle tüm devreler MSI'dan LSI 'ye(Geniç ölçekli tümleştirme) geçmeye başladılar. MSI ve LSI arasındaki sınır net olarak tarif edilmemiştir, ancak LSI çipler bir önceki jenerasyona göre birkaç kat daha fazla transistör içerir.

Intel 1978 yılında 8086 CPU çipini üretti bu çip yaklaşık 29000 transistör içermekte ve harici (çipin dışında) vs dahili olarak 16 bitlik veri yolları kullanıldı. Bu çipin PC teknolojisindeki yeri çok önemlidir, çünkü CPU çiplerindeki evrim içinde daha sonra geliştirilen tüm Intel CPU'lar mimari nüvelerinde 8086 özelliklerini taşırlar. Yani 8086 uyumlu konumları (veya kipleri) içerirler.

Intelin 1979 yılında ürettiği 8088 CPU çipi ilk çıkan PC lerin esasını teşkil etti. Bu çipin iç mimarisi 8086'nin aynısıdır, sadece dış veri yolu 16 yerine 8 bit'ten oluşur.

Yine 1979 da Motorola'nın ürettiği 68000 (yaklaşık 68000 transistör) CPU çipi Apple, Amiga, Atari gibi IBM uyumlu olmayan kişisel bilgisayarların esasını oluşturdu.

CPU evrimindeki gelişmeler 1982'de 80286 (134,000 transistör), 1985'te 80386 (275,000 transistör), 1988'de 3865X (275,000 transistör), Genelde 100,000 ve daha yukarı sayıda transistör içeren tümleştirme teknolojisine VLSI (Very Large Scale Integration- Çok Geniş Ölçekli Tümleştirme) olarak adlandırılır.

1989'da 486 (1,200,000 transistör) 1990'da 3865L (855,000 transistor), 1991 de 4865X (1,185,000) transistör ve nihayet 1993'te Pentium'un (daha önce 586 olarak adlandırılmıştı) (3,100,000 transistör) üretilmesiyle devam etti. 1 milyon ve üzeri transistör içeren tümleştirme teknolojisi ise ULSI (Ultra Large Scale Integration- Ultra Geniş Ölçekli Tümleştirme) olarak adlandırılır.

Aynı yıllarda Motorola firmasında eş performansa sahip 68010, 68020, 68030, ve 68040 (486 performans eşdeğeri) CPU çiplerini üretti.

İleriye doğru bugünden tahmin edilebilen gelişmeler ise 1999-2001 yılarında 786 (100,000,000 transistör) CPU çiplerinin üretilebileceğidir. Bu çiplerin gerçekte alacakları isimler pazara çıktıklarında belli olacaktır.

2.2.2 PC-Bilgisayarların Evrimi

İlk kişisel bilgisayar IBM tarafından Ağustos 1981'de açıklandı. İlk PC: 4.77 MHz de çalışan bir 8088 CPU; 64KByte Bellek; bir adet 5.25-inch ve tek taraflı 160KB kapasiteli disketleri kullanan sürücü;harici veri saklama birimi olarak kullanılabilir bir kaset teyp için arabirim; ve yüksek kalitede yazı gösterebilen tek renk (monokrom) veya düşük kalitede yazı ve grafik gösterebilen renkli bir ekran (monitör) içeriyordu.

1983'te çıkan PC-XT (extended technology) de ilk defa sabit disk kullanabilme yeteneği eklenmişti. Başlangıçta ek ücrete tabi olan sabit diskler (fixed disk veya hard disk) giderek kişisel bilgisayarların standart donanımı haline geldi.

Daha sonra XT bilgisayarlarında "Turbo" denilen daha hızlı (8, 10, 12 Mhz) 8088 çipleri kullanılmaya başlandı.

1984 Ağustosunda çıkan PC-AT (Advanced Technology) bilgisayarlar ise 80286 (veya kısaca 286) CPU çipini kullanmaya başladılar. Bu sayede 16 bitlik veri yolları kullanılmaya başlandı ve bilgisayarların performansı 8 bitlik veri yoluna göre iki kat arttırılmış oldu. AT tipi bilgisayarlar, yarı iletken üretim teknolojisindeki gelişmeler sayesinde, giderek artan (12, 16 ve 20 MHz) hızlarda çalışabilen 286 CPU çipleriyle donatıldılar.

AT tipi bilgisayarlardaki diğer bir hız kazancı da, 286'da kullanılan azaltılmış komut işleme süresiydi. Böylece 5 MHz'de çalışan bir 8086'ya göre, 16 MHz'de 286'nin verimi (throughput) 13.3 katı daha fazladır. CPU çiplerinin işlem hızı MIPS (Million Instructions per Second- Saniyede Milyon Komut) olarak gösterilir.

AT tipi bilgisayarlar IBM için önemli bir başarı oldu. 1984-1987' ye kadar IBM, AT tipindeki ürünlerinde büyük değişiklikler yapmadı. Yeni çıkan modeller ise diğerlerine göre küçük farklılıklar içeriyordu.

1987'den başlayarak IBM yeni bir kişisel bilgisayarlar serisi üretmeye başladı. PS/2 veya Personal System/2. Bu dönemden başlayarak IBM kendi ürettiği kişisel bilgisayar sistem modellerini isim vermek yerine numaralarla belirtmeyi tercih etti. PS/2 model 30, 50, 60, 80 gibi. Bu model numaraları bilgisayarın kullandığı CPU çipine ve diğer donanımına göre değişiyordu.

Örnek olarak PS/2 model 25, 30, ve 50, 286 CPU, PS/2 model 355X, 40SX, 555X, 3865X CPU, PS/2 model 70 ve 80, 386 CPU ve PS/2 model 76 ve 90, 486 CPU çipini kullanırlar. Doğal olarak burada belirtilen kısıtlı örneklerin dışında da değişik IBM modelleri üretildi. Teknolojik gelişmelere ve pazar gereksinimlerine uygun olarak yeni modellerde sürekli olarak piyasaya çıkarılmaktadır.

PS/2 modelleriyle birlikte IBM dört yeni teknoloji standardını ortaya çıkardı:

1. MCA (Micro Channel Architecture) iletişim yolu
2. VGA (Video Graphics Array) ekran gösterim (display) standardı
3. 3.5-inch disket sürücüsü (5.25-inch'liklerin yerine)
4. 101 tuşlu geliştirilmiş klavye (Bu klavye daha önce de var olmakla birlikte aynı dönemde farklı klavye tipleri de kullanılıyordu).

2.2.3 IBM Uyumlu Bilgisayarlar (Personal Computer Clones)

1981'de IBM ilk kişisel bilgisayarını (PC) çıkarmadan önce ABD'de bir çok firma, daha önce bahsettiğimiz 8 bitlik CPU çiplerini (Motorola-6800, Mostek-6510, Intel8080, Zilog-Z80 vs.) kullanan kişisel bilgisayarlar (Apple II, Tandy TRS-80, Commodore PET, vs.) üretiyorlardı. Ancak aralarında belli bir standardın oluşmamış olması (hepsi birbirlerine uyumsuz işletim sistemleri ve programlarla çalışıyorlardı) ayrıca arkalarında güçlü bir firma (IBM gibi) desteği olmaması bu bilgisayarların yaygınlaşmasını ve pazar hacminin büyümesini önüyordu.

IBM PC'nin baştan itibaren piyasada gösterdiği başarı, (talep hemen arzın üzerinde oluşmuştu) bir çok kişinin dikkatini PC piyasasının genişleme potansiyeline çekti. Diğer taraftan PC'de kullanılan başta CPU ve diğer çiplerin patentlerinin IBM'e ait olmaması ve Intel, Texas, Motorola gibi diğer firmalar tarafından üretilmesi, bu çiplerin piyasada serbestçe satılmasına olanak tanıyordu. Aynı şekilde işletim sisteminin (DOS) patenti de Microsoft firmasına aitti.

IBM PC piyasasında tekel olma gayreti içinde, tasarım mimarisi ve bilhassa iletişim yolu ile ilgili teknik bilgileri açık olarak piyasaya sunmuş ve patent hakkı talep etmemişti. Bu davranışın amacı üçüncü şahısları PC için yeni fonksiyonlar içeren kartlar üretmeye yönlendirmektir.

Çünkü PC içindeki genişleme yuvalarının yarattığı, ilave fonksiyonlar kazanma potansiyeli, pazarda IBM PC'nin tercih edilmesini sağlayan önemli bir öğeydi.

PC'yi cazip kılan diğer iki öğe ise, birincisi işletim sistemi içinde ücretsiz olarak verilen ve kullanıcının kendi programlarını yaratmasını sağlayan Basic interpreter programı, diğeri de daha önce var olan birçok IBM programının PC üzerinde de çalışabilme olasılığıydı. Her ne kadar bu programların pek azı kişisel kullanıcının gereksinimlerini karşılıyor olsa da müşterilere cazip geliyordu.

Yukarıda anılan nedenler kısa süre içinde PC ve daha sonra XT ve AT'nin diğer firmalar tarafından işlevsel olarak taklit edilmesine neden oldu (Clone). Birebir taklit IBM'e özgün tasarımın patent hakkını ihlal oluyordu. Doğal olarak diğer firmalardan temin edilebilen çiplerin aynı işlevleri yapabilecek (yani MSDOS'u çalıştırabilecek) şekilde irtibatlandırılmasının birden fazla metodu vardı. (İşlevsel eşleniklik- Emülasyon).

İşte bu nedenlerin tümü bütün dünyada çok geniş bir PC uyumlu bilgisayar piyasası yarattı. O kadar ki artık satılan kişisel bilgisayarların büyük çoğunluğu IBM yerine başka firmalardan gelmektedir.

Piyasadaki keskin rekabet ise fiyatların sürekli kırılmasına ve bilgisayar donanım bileşenlerinin çoğunun üretiminin (Maliyetlerin daha ucuz olduğu) Uzak Doğu ülkelerine (Tayvan, Hong-Kong, Kore vs) kaymasına neden olmuştur. Zaten son yıllarda Intel CPU çiplerinin de diğer firmalarca içlevce benzerleri yapılabildi (emulation) ve CPU çip fiyatları önemli ölçüde düştü. Böylece her ülkede; bileşenleri ayrıık olarak ithal edilmiş kişisel bilgisayar parçalarını bir araya getirerek, bilgisayar üretip ucuz fiyatlarla kullanıcıya sunma imkanı doğdu.

2.2.4 IBM'le yol ayrımı ve Endüstri Standardı bilgisayarlar

Başlangıçta kişisel bilgisayar piyasasına hakim olan IBM artan rekabet karşısında kendisini giderek küçülen bir pazar payı ve azalan kârlarla karşı karşıya buldu. Bu durumdan kurtulmak için 1987 yılında IBM Kişisel bilgisayarların performansını önemli ölçüde arttıran ve teknik detaylarını ancak ücret karşılığı vereceğini açıkladığı Mikro Kanal Mimarisini (MCA-Micro Channel Architecture) uygulamaya koydu. Dahası IBM bu teknolojiyi satın almak isteyenlerden, daha önceki AT teknolojisini kullandıkları için ayrıca hak talep etmeye başladı. Bu gelişmelere tepki gösteren diğer Kişisel Bilgisayar donanım ve yazılım üreticileri bir araya gelerek Endüstri Standardı Mimarisi Kurumunu ((ISA- Industry Standards Architecture) Association) kurdular. Böylece IBM'le topluca rekabet edilebilecek ve bilgisayar performansını MCA seviyesine arttırıcı teknolojik gelişmeler herkesle paylaşılabilirdi.

Nitekim 1987 yılına kadar oluşmuş olan AT-ISA mimarisi kullanıcı tabanı (milyonlarca bilgisayar) ISA standardının devamını sağladı, MCA'ya karşı EISA mimarisini çıkardı ve Vesa local bus gelişmelerini oluşturdu. O günlerde en yaygın olarak ISA ve devami olan Vesa local bus kullanıldı. Ancak 1994'ün sonundan başlayarak PCI standardına doğru bir geçiş başladı.

IBM ise MCA ile umduğu sonucu alamadı sadece IBM kullanmaya alışık kurumsal müşteriler (şirketler, devlet kuruluşları) satış sonrası hizmetlerde ,süratli ve güvenli servis istedikleri için giderek azalan miktarda IBM kişisel bilgisayarları kullanmaya devam ettiler.

Günümüzde artık kişisel bilgisayarın IBM uyumlu olmasından ziyade, Endüstri standardı olup olmadığı daha doğrusu MS-DOS ve bu işletim sistemi için yazılan programlara uyumlu olup olmadığı daha önemli olmuştur.

Kişisel bilgisayarların diğer uyumluluk kriterleri: A) Kullanılan iletişim yolu tipi. B) kullanılan CPU tipi ve ,C) kullanılan gösterim adaptörü ve monitörün (ekranın) tipidir.

2.2.5 Kişisel Bilgisayarların Bileşenleri

Bir kişisel bilgisayar beş ana işlevsel bileşenden oluşur. Bunlar

1. İşlem birimi (main processor veya kısaca CPU),
2. Bellek birimleri,
3. Giriş/Çıkış (I/O) devreleri,
4. Veri saklamak için disk(ler) ve son olarak
5. Programlar

İşlevsel bileşenler dışında destek bileşenleri de mevcuttur. Bunlar bilgisayar işlevsel bileşenleri içinde bulunduran 1."Kutu" veya ("Kasa"), 2.güç kaynağı, ve 3.tüm elektronik devre elemanlarını bu elemanlar arasındaki elektriksel bağlantıları, genişleme yuvalarını (bu genişleme yuvalarındaki I/O, video, ve diğer kartları) içeren anakart (veya plaket) tir.

Herhangi tip bir bilgisayar ne kadar çok yetenekli ve hızlı çiplerle donatılmış olursa olsun, bu donanım üzerinde çalışacak bir program olmazsa o bilgisayar hiç bir şey yapamaz. Programlar iki ayrı bölümde toplanabilirler: Sistem programları ve Uygulama programları. Tüm programlar belirli işlevleri gerçekleştirirler. Sistem programları bilgisayarın işletilmesine, diğer bir deyişle görevlerini yerine getirebilmesine yardımcı olurlar. Gerçekten bilgisayarın dahili işlemleri o kadar karmaşıktır ki, bu işlemler sistem programlarının yardımı olmadan çalışmazlar.

Uygulama programı ise kullanıcının yapmak istediği belli bir görevi yerine getirir. Bir bordro hazırlama programı, bir stok kontrol programı, bir kelime işlem programı bu tip programların tipik örnekleridir.

Bazı sistem programları sürekli olarak bilgisayarın içinde saklanır. Bu programlar silinemeyen bellek birimi (ROM) içinde kayıtlıdır dolayısıyla bilgisayar kapatıldığında kaybolmazlar. Bu ROM tabanlı programlar en temel (ve en alt düzey) denetim ve destek işlemlerini yaparlar, ayrıca tüm uygulama programlarının gereksinim duyduğu belli hizmetleri karşılarlar.

Bu hizmet programlarına BIOS (Basic Input/ Output Services Temel Giriş/ Çıkış Hizmetleri) denilir. Services ve System (Hizmetler veya Sistem) literatürde birbirinin yerine kullanılmaktadır. Esasen aynı anlamda kullanılırlar. Yani "Hizmetlerin" tümü bir "sistem" oluşturur.

Diğer sistem programları ise BIOS "sisteminin" oluşturduğu temel üzerine yapılandırılmış olup daha üst düzey destek hizmetleri verirler. MS-DOS gibi işletim sistemleri bu programların örnekleridir.(OS/2, UNIX). Bu işletim sistemleri PC tipi bilgisayarlarda, bilgisayar donanımının içinde saklanmayıp, disk veya disket üzerinden, (ilk çalıştırma işlemlerinde-power on) BIOS tarafından bellek (RAM) birimine yüklenirler.

Ana kart veya Sistem Kartı (Mother Board) Merkezi işlem birimini (CPU), bellek çiplerini, CPU destek işlevleri çiplerini, Giriş/Çıkış (I/O) arabirimlerini (seri-port, paralel- port, klavye arabirimi, disk arabirimi, vs), veriyolları ve genişleme yuvalarını üzerinde taşır.

Giriş/Çıkış (I/O) bilgisayarın verileri içine alma ve sonuçları tekrar dışarı gönderme yöntemine verilen isimdir. Klavyeden veya Mouse (Fare) den girilen veriler, Printer veya Ekrana gönderilen veriler hep I/O işlevlerini kullanırlar. Verilerin girdi/çıkışı için kullanılan bu birimlere I/O birimleri (device) denir. Aynı zamanda çevre (peripheral) birimleri olarak ta adlandırılırlar.

Diğer bir I/O birimi ise diskdir (sabit ve/veya disket) Bu birim diğer I/O birimleri gibi kullanıcıya açık olmayıp sadece bilgisayarın kullanımı içindir. BU birim bilgisayarın kütüphanesi gibi olup, her tür programı (İşletim sistemi DOS dahil) sürekli (ve süresiz) saklamak içindir. Burada saklanan veriler bilgisayar kapatıldığında kaybolmaz. çünkü manyetik ortam üzerine kaydedilmişlerdir.

Bilgisayar her hangi bir programa gereksinim duyduğu zaman ilgili programı diskten (veya disketten) bellek birimine çeker ve buradan okuduğu komutlarla yine buradan okuduğu veriler üzerinde işlemler yapar, ve sonucu yine bu bellek birimine kaydeder.

2.2.6 Ana Donanım Bileşenleri (Main Hardware Components)

Ana Kart (veya Sistem Kartı).

Daha önce de kısaca değindiğim gibi bilgisayarın ana işlevsel birimlerini (disk sürücülerini hariç) taşır. Bunlar arasındaki elektriksel irtibatları üzerindeki iletken yollarla (baskı devre) sağlar.

Ana kart üzerinde:

- Merkezi İşlem Birimi entegresi CPU (bilgisayarın modeline göre soketli veya lehimli)
- CPU tipine göre Matematiksel işlemci entegresi (soketli)
- "Overdrive" Entegresi (Soketli) (Yazılımın türüne göre sistem verimini artırır ancak bulunması zorunlu değildir)
- Diğer yardımcı işlemler için gerekli entegreler
- BIOS ROM(ları)
- SIMM Bellek Birimleri ve bunların takıldığı soketler
- "Cache" bellek entegreleri
- Disket ve disk sürücülerinden gelen konnektörlerin takıldığı soketler
- Güç kaynağından gelen konnektör için soket
- Klavye soketi
- Çevre birimleri için ara birimler (entegreler)
- "Gerçek zaman saati" entegresi, saat kristali ve pili.
- Donanım yapısını (configuration) değiştirebilen anahtar (DIPswitch) veya atlama bağlantılar (jumpers)
- Bilgisayara ek yetenekler kazandıran (Modem, Fax, Ses, Video, Harici Seri ve Paralel port, vs) ilave kartların takıldığı genişleme yuvaları (soketler) dizisi
- Genişleme yuvalarını birbiriyle ve bilgisayarın diğer kısımlarıyla iletişimini sağlayan yol sistemi (BUS)

Yukarıda belirtilen seri ve paralel portlar, mouse ve joystick portları bazı ve bilhassa yeni modellerde, genişleme yuvalarından birine takılan port kartı üzerine toplanmış olabilir.

2.2.7 BUS (YOL.) Anlamı, geređi ve farklı "yol" standartları:

PC uyumlu bilgisayarlarada adaptörler tasarlayan üreticiler ve kullanıcılar için "YOL" çok önemlidir. Bu adaptörler daha önce de belirtildiđi gibi, bilgisayara ilave yetenekler kazandıran. fax, modem, ses, vs. gibi kartlardır. Bu kartlar kullanıcının istek ve tercihine göre ilk başta veya sonradan takılabilirler ve ek ücrete tabidirler.

Adaptörlerin her cinsinin bilgisayarla kolaylıkla irtibatlandırılabilmesi ve hepsinin aynı "elektronik yöntemle" bilgisayarın diđer kısımlarıyla iletişim kurabilmesi uyumluluk için gereklidir. Bu olanakla geniş bir pazar hacmi yaratılır ve adaptör birim maliyeti ve dolayısıyla fiyatı da en düşük olası değere iner. (Diđer ticari unsurlar hariç olmak üzere).

Birden fazla ve birbirinden bağımsız elektriksel iletişim hattının bir arada (paralel hatlar üzerinden) gönderilmesi yöntemi BUS (YOL) olarak isimlendirilir.

Ana kartta "YOL" üzerine monte edilmiş olan genişleme yuvaları, yani konnektörler, adaptör kartları ve "YOL" arasındaki iletişimi sağlarlar.

PC ve uyumlu bilgisayarların evrimi sürecinde birden fazla BUS standardı oluşmuştur.

En yaygın olarak kullanılan BUS, ISA (= AT) "yol"udur. ISA (Industry Standard Architecture) yolu aynı zamanda daha önceki nesil olan XT yolunu da içerir. Yani XT uyumlu adaptörler ISA yoluna takılabilir, ancak tersi olanaklı değildir. XT yolu özet olarak 8 bit veri (data) yolu, 20 bit adres yolu (1MByte). çeşitli kontrol işaretleri, +12, -12, +5, -5 volt seviyelerinde besleme hatları, saat (clock) hattı, ve toprak hatlarını içeren 62 hatlık (ve dolayısıyla 31 pozisyonlu ve 62 bacaklı konnektör kullanan) yol standardıdır.

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

8 Bit ISA (XT) yolu bacak bağlantıları

<u>Bacak No:</u>	<u>İşlevi</u>	<u>Bacak NO:</u>	<u>İşlevi</u>
A1	I/O CH CK	B1	Ground
A2	D7	B2	Reset Drive
A3	D6	B3	+5V DC
A4	D5	B4	IRQ9
A5	D4	B5	-5V DC
A6	D3	B6	DRQ2
A7	D2	B7	-12V DC
A8	D1	B8	WS
A9	DO	B9	+12V DC
A10	I/O CH RDY	B10	Ground
A11	AEN	B11	SMEMW
A12	A19	B12	SMEMR
A13	A18	B13	IO W
A14	A17	B14	IO R
A15	A16	B15	DACK3
A16	A15	B16	DRQ3
A17	A14	B17	DACK1
A18	A13	B18	DRQ1
A19	A12	B19	Refresh.
A20	A11	B20	CLK
A21	A10	B21	IRQ7
A22	A9	B22	IRQ6
A23	A8	B23	IRQ5
A24	A7	B24	IRQ4
A25	A6	B25	IRQ3
A26	A5	B26	DACK2
A27	A4	B27	T/C
A28	A3	B28	BALE
A29	A2	B29	+5V DC
A30	A1	B30	OSC
A31	AO	B31	Ground

16 Bit ISA (AT) yolu bacak bağlantıları

<u>Bacak No: İşlevi</u>		<u>Bacak No: İşlevi</u>	
A1	I/O CH CK	B1	Ground
A2	D7	B2	. Reset Drive
A3	D6	B3	+5V DC
A4	D5	B4	IRQ9
A5	D4	B5	-SV DC
A6	D3	B6	DRQ2
A7	D2	B7	-12V DC
A8	D1	B8	WS
A9	DO	B9	+12V DC
A10	I/O CH RDY	B10	Ground
A11	AEN	B11	SMEMW
A12	A19	B12	SMEMR
A13	A18	B13	IO W
A14	A17	B14	(O R
A15	A16	B15	DACK3
A16	A15	B16	DRQ3
A17	A14	B17	DACK1
A18	A13	B18	D RQ1
A19	A12	B19	Refresh
A20	A11	B20	CLK
A21	A10	B21	IRQ7
A22	A9	B22	IRQ6
A23	AS	B23	IRQ5
A24	A7	B24	IRQ4
A25	A6	B25	IRQ3
A26	A5	B26	DACK2
A27	A4	B27	T/C
A28	A3	B28	BALE
A29	A2	B29	+5V DC
A30	A1	B30	OSC
A31	AO	B31	Ground
C1	BHE	D1	MEM CS16
C2	LA23	D2	I/OCS16
C3	LA22	D3	IRQ10
C4	LA21	D4	IRQ11
C5	LA20	D5	IRQ12
C6	LAI9	D6	IRQ15
C7	LAI8	D7	IRQ14
C8	LAI7	D8	DACKIO
C9	MEMR	D9	DRQO
C10	MEMW	D10	DACK5
C11	DS	D11	DRQ5
C12	D9	D12	DACK6
C13	D10	D13	DRQ6
C14	D11	D14	DACK7
C15	D12	D15	DRQ7
C16	D13	D16	+5V DC
C17	D14	D17	Master
C18	D15	D18	Ground

MCA-Micro Channel Architecture (Mikro Kanal Mimarisi)

IBM tarafından PS/2 sistemleri için geliştirildi. İletişim hızı 20 MB/saniye'ye kadar çıkabilen bu yol genellikle işaretleşme hatları topluluğundan oluşur. Her bir hattın amacı ve elektriksel işaretlerin zamanlama bağıntıları "yol"un tarifini oluşturur. Kanal ise belirli bir yolu referans alıp, bu yol üzerinde veri iletişimi ve kontrol protokolünü tarif eder. Dolayısıyla MCA farklı "yol"ların temel olarak aldığı karmaşık teknik nitelikleri içerir.

MCA "yol"unun 16 bit ve 32 bit olan iki değişik şekli vardır. '16 bit MCA adaptörlerinin 116 bağlantı ucu vardır. Bunlardan 77'si işaretleşme hattı, 12 güç besleme hattı, 17 toprak hattı, 1 audio toprak hattı, 5 rezerve edilmiş hat ve 4 anahtarlama konumu. Topraklama hatları işaretleşme hatları arasına eşit şekilde dağıtılmış olup hatlar arasındaki girişi asgariye indirirler. Güç besleme hatları ise +12, -12 ve +5 volttan her hangi birini taşırlar. 77 işaretleşme hattının 24'ü adres 16'sı ise veri hattıdır. Geri kalan hatlar ise kontrol işaretlerini içerir. 16 bitlik MCA "yolu" 286 CPU'lar için tasarlanmıştır

Intel 386, ve 486 gibi 32 bitlik veri yolu olan CPU lar içinse 32 bitlik MCA "yolu" tasarlandı. Bu yol toplam 186 ayrı hat içerir ve 93 pozisyonlu konnektör kullanır.

MCA "yol"unun başlıca avantajları şunlardır:

1. MCA adaptörleri çok daha az elektriksel girişime (enterferans) neden olurlar. Elektriksel girişim veri kaybı, veri bozulmaları ve hatalarında önemli bir etkidir.
2. MCA yolu adaptörlerden gelen iletişim taleplerini (interrupt) daha yüksek performansta yanıtlayabilir, böylece veri güvenilirliği artar ve veri kaybı azalır.
3. MCA, BUS master (Yol denetleyicisi) denilen özel adaptörlere olanak tanır.

Bu tip adaptörler üzerlerinde bulunan prosessörler (işlemci) yardımıyla bir çok işlevi, CPU'nun katkısı olmadan kendileri yapabilirler, ve merkezi işlemci bu arada başka görevleriyle ilgilenir.

4. MCA yolunu kullanan adaptörler kendilerini tanımlıyabilirler.
5. MCA yolunu kullanan adaptörler bozulurlarsa, bu kartların işlevlerini düzenli aralıklarla denetleyen bir program tarafından kapatılabilirler.

EISA (Extended ISA- Uzatılmış (Genişletilmiş) ISA) yolu.

Bazı literatürde Enhanced-Geliştirilmiş ISA olarak ta adlandırılmış olan EISA yolu 32 bitlik CPU'lar için tasarlanmış olup 16 bit'lik ISA yolunun genişletilmiş şeklidir. EISA yolu MCA yolu performansma eşdeğer bir performansa ulaşmakla kalmayıp aynı zamanda ISA tipi adaptör kartlarına da uyumludur. İlk olarak 1988 yılında IBM dışındaki PC üreticileri tarafından tanıtıldı: Esas amacı ise daha önce bahsettiğim ticari sebeplerden dolayı 1987 yılında IBM tarafından tanıtılan MCA "yol"una rekabet edebilecek bir alternatif oluşturmaktır. EISA uyumlu adaptör kartları ancak, EISA'nın tanıtımından iki yıl sonra ortaya çıktı ve EISA yolu piyasada yaygın şekilde kabul görmedi.

Ancak EISA gerçek amacı olan MCA yolunun yaygınlaşip yeni bir standart olmasını önleme konusunda başarılı oldu.

Yerel Yol (Local Bus)

PC'lerde ilk başlarda BUS, çevre birimleri yanında bellek, disk ve monitör (ekran) giriş/çıkışı için de kullanılıyordu. Yeni "yol" tasarımlarının temel amaçlarından biri de bilhassa ekrana aktarılan verilerin, bellekle ve diskle olan, iletişim hızını artırmaktır. Veri hattının 16 bit'ten 32 bit'e çıkarılması iletişimi hızlandırmakla birlikte yine de "yol" hızıyla (MHz) sınırlı kaldı.

Öncelikle, bellekle iletişim hızı, PC üreticilerinin bellek birim(ler)ini ana kart üzerine taşımaları ve CPU ile iletişimi "yol" üzerinden sağlamalarıyla sistem hızına çıktı (25 veya 33 MHz)

VESA VL BUS - VESA Local Bus (VESA Yerel yol)

Video Electronics Standards Association (Video Elektronigi Standartları Kurumu) Aralık 1991 de kuruldu. İlk standardı olan VL bus 1.0 Ağustos 1992 de yayınlandı. Bu standartla birlikte yukarıda bahsettiğimiz yerel yol "uzatılarak" ISA veya EISA yoluna paralel bir yol haline getirildi. CSyleki VL Bus donanımlı bilgisayarlar hem ISA (veya EISA) hem de VL Bus uyumlu kartları birlikte kullanabilmektedirler. Tabi ki sadece ISA (veya EISA) uyumlu kartlar, VL bus olanaklarından yararlanamaz ancak yan yana bulunan genişleme yuvalarına takılabilirler.

PCI Yerel Yolu (Peripheral Connect Interface-Çevre bağlantı arabirimi)

PCI diğer bir yerel yol standartıdır, VESA yoluna eşdeğer olanaklar sağlamaktadır. Diğer bir özelliği hem +5 volt hemde, özellikle Dizüstü ve "Notebook" tipi bilgisayarlarda yerleşmekte olan +3.3 volt iletişim standartlarına uyumludur. Bu "yol"da 32 veya 64 bit'lik veri yollarını desteklemektedir. Veri iletişim hızı 32 bit'te 132 Mbyte/saniye, 64 bit'te ise 264 MByte/Saniye olarak açıklanmıştır.

2.2.8 CPU

Genelde PC olarak tanımladığımız kişisel bilgisayarlar gerçekte çeşitli nesilleri olan bir soyağacı oluştururlar. Bu olguya PC ailesi de diyebiliriz. Tüm PC'ler Intel CPU veya "uyumlu" çipler temelinde tasarlanmıştır.

PC üzerinde Merkezi İşlem Birimi (CPU) daima bir tanedir. Bu işlemci de daima Intel 86 tipi veya ona "uyumlu" bir işlemcidir. Artık bir çok firma "Uyumlu" işlemciler üretmektedir. Bunlar arasında; Texas Instruments, Advanced Micro Devices (AMD) ve Cyrix'i sayabiliriz.

"Uyumlu" işlemciler genellikle orijinallerine göre ek gelişmeler (enhancement) içerirler. Bunun sebebi ise daha sonra tasarlandıkları için orijinalden elde edilmiş olan deneyimlerden yararlanmış olmalarıdır. Kullanıcı açısından teknik olarak Intel veya "Uyumlu" CPU arasında bir fark yoktur. (Özel performans ölçücü programlar "Uyumlu" CPU'ları verimlilikte (throughput) daha üstün bulabilirler)

Uyumlu işlemcilerin kullanıcı üzerindeki en önemli etkileri ise yarattıkları rekabet unsurundan dolayı CPU fiyatlarındaki sürekli ve önemli düşmelerdir. Bazı PC'ler yardımcı işlemciler (coprocessor) içerir (486 öncesi nesiller). Bu işlemcilere "Kayan nokta-Floating Point" işlemci de denilir. Bu tip işlemciler Intel '87 tipi işlemcilerdir. Görevleri ise özel matematiksel ve trigonometrik hesapları çok hızlı şekilde yapabilmektir. Merkezi işlem birimleri de aynı hesaplamaları yazılım programları yardımıyla yapabilirler ancak bu tip işlemlerde yardımcı işlemcilere göre çok daha yavaşlardır.

Intel'in ilk 16 bit veri yollu işlemcisi 8086 Haziran1978'de tanıtıldı. Haziran 1979' da ise 8086'nin işlevsel eşdeğeri olan ancak harici olarak 8 bit'lik veri yolu kullanan 8088 tanıtıldı.

8088, 1981 de tanıtılan ilk PC nin tabanını oluşturdu.(IBM (ve diğer firmalar) daha sonra 8086 tabanlı bazı PC modelleri de üretti) İlk PC'nin 8088 tabanlı olmasının iki sebebi vardı. Birincisi çok miktarda üretildiği için birim fiyatı daha ucuzdu, ikincisi de o dönemlerde 16 bit veri yolunu kullanabilecek diğer yardımcı çipler ve bellekler henüz çok miktarda üretilmedikleri için yine çok pahalıydı.

Daha sonra Şubat 1982'de tanıtılan 80286, 8086 tasarımına ilave yetenekler kazandırdı. Ekim 1985' te tanıtılan 386 ise 286' nin üstüne, Nisan 1989'da tanıtılan 486 ise 386 üzerine yapılan geliştirmelerle elde edildiler. Bu şekilde Intel serisi işlemciler geriye dönüşümlü olarak uyumludurlar. Diğer bir deyişle 8086 dan başlayarak her nesil bilgisayar için yazılmış olan programlar daha sonraki nesil CPU' lar üzerinde de çalışabilir. (Ancak tersi olası değildir).

Teknik olarak 486,386,286 işlemcileri 8086 uyumlu konumları (veya kipleri) içerirler.

Intel firması niçin uyumluluk üzerinde bu kadar durmuştur? Uyumluluğu sağlamak demek olası diğer teknolojik gelişmelerden kısmen de olsa vazgeçmek demektir.

Bunun sebebi ise tamamen ticaridir. Çünkü her nesil bir önceki ve bilhassa 8086(8088) için yazılmış olan programları çalıştıramazsa, bu (çoğu da gayet iyi iş gören) yazılımlar için yapılmış yatırımların boşa harcanması demektir. Müşteriye ise en azından yeni nesil programlara geçebilmesi için zaman tanımak gerekir. Yani bu süre içinde müşteri yeni yazılımlar için gerekli yatırım fonlarını oluşturacaktır.

Aksi halde müşteriler ve bilgisayar üreticileri farklı CPU tabanlı (veya platformu) kullanmayı tercih edebilirlerdi ve PC'ler bugünkü dünya standartını oluşturamazlardı.

Intel CPU'larının genel özellikleri

Bu alt bölümde Intel serisi bilgisayarların genel özelliklerini ve birbirlerine göre farklılıklarını inceleyeceğim.

8086 Intelin ürettiği ilk harici ve dahili (CPU çipinin içinde) 16 bit veri yolu kullanan CPU'dur. 40 bacaklı DIP (Dual inline pins) ve 44 bacaklı PLCC (Plastic Leaded Chip Carrier) entegre paketlenmesi şekillerinde bulunur. 5, 8,10 MHz saat frekanslarında çalışan tipleri de mevcuttur. 16 bacağı adres ve veriler için ortak olarak kullanılır.

Bu yöntem Multiplexing-Çoğullama denilir. Adres ve Veriler birbirlerinden ortak yol üzerinde AD0-AD15 bulunan diğer bir entegre tarafından ve CPU'dan gelen ALE (Address Latch Enable) işaretiyle birbirinden ayrılıp, A₀-A15 adres bitleri A16-A19 bitleriyle aynı adres yoluna gönderilir. Toplam 20 bit'lik adres yolu ile 1 Mbyte'lık ($2^{20}= 1.048.576$ Byte) adrese ulaşabilir.

NEC firmasında 8088'e uyumlu V20 ve 8086'ya uyumlu V30 CPU'ları üretmiştir. Orijinal CPU'larına göre %30 daha verimli olan bu işlemciler bilhassa o dönemdeki Japon kaynaklı bilgisayarlarda kullanıldılar.

80286 işlemcisiyle Intel gerçekten "bir sonraki nesil" işlemciyi gerçekleştirdi. Bu işlemcinin en önemli özellikleri: (8086 kipinde çalışabilme yeteneğine ek olarak)

1. Genişletilmiş bellek adresleme olanaklarıyla (24 bit) 16Mbyte'a kadar bellek adresliyebilmesi ($2^{24}=16.777.216$ Byte);

2.1 Gbyte'a (1 GB=1000 MB) kadar sanal bellek (virtual memory) simülasyonu yapabilmesi (disk belleği, dahili (RAM) belleğin uzantısı gibi kullanabilmesi)

3. Donanım çok görevliliğine (Hardware multitasking) sahip olması ki bu sayede işlemci çeşitli (ve farklı) görevler (task) arasında süratle gidip gelebilmesi, olarak sayabiliriz.

Çok görevlilik olanağı aynı anda birden fazla programı işletme olanağı sağlar. Bunun göstergesi ise ekranda her biri farklı program içeren birden fazla "pencerenin" aynı anda açılabilmesidir. Daha önceki bilgisayarlarda da çok görevlilik yazılım yardımıyla olası idiye de, donanım desteği olmadan bu tür işlevler güvenli olmayıp, işletme esnasında programların bir şekilde bozulup durmasına (crash) sebep oluyordular.

Aslında aynı anda çalışıyor gibi görünen iki farklı program arasında CPU işlevini bölüştürmekte ve ikisi (veya daha fazlası) arasında süratle gidip gelmektedir.

Diğer bir 286 avantajı da, bu nesil işlemcide 6MHz'de başlayan çalışma hızının yeni versiyonlarla 20 MHz'e kadar çıkmasıdır. 286 çok görevliliği korumalı kipte (protected mode) sağlar. Bu konumda 16 MB belleğin tümünü adresleyebilir. Normal konumda ise 8086 gibi çalışır ve sadece 1 MB bellek adresleyebilir.

Ancak 286 döneminde kullanıcıların büyük çoğunluğu MS-DOS işletim sistemini kullanıyorlardı. DOS ise tek görevli bir işletim sistemi olduğu için 286'nin sunduğu yeni olanaklardan yararlanamıyordu.

Sonuç olarak ta 286 daha önceki CPU'ların çok hızlı bir modeli gibi çalışıyordu. Buna rağmen bu dönemde OS/2 ve UNIX işletim sistemlerine sahip olan kullanıcılar 286'nin yeni olanaklarından yararlanabildiler.

Bu tezin konusu dışında kalan 386, 486 ve Pentium gibi gelişmiş CPU modellerinin iç yapısını anlatmadan CPU donanımı konusunu burada keseceğim.

CPU ve harici birimler arasındaki haberleşme yöntemleri.

CPU'lar kendi dışlarındaki "dünya" ile üç farklı yöntemle haberleşebilirler. Birincisi yardımcı işlemci ile PEREQ ve PEACK işaretleriyle haberleşir. Diğer ikisi ise bellek ve port'ların kullanımınıdır.

CPU kendi içinde yaptığı işlemler için bellekle haberleşir. Bellek, CPU'nun çalışma alanı olarak düşünülebilir. CPU'nun komutları ve bu komutlarla işlenen veriler bellekte kayıtlıdır. Sonuçlar da yine belleğe kaydedilir.

"Port" yani "kapı" CPU'nun bellek dışındaki birimlerle konuşma yöntemidir. CPU'nun bilgisayar içinde iletişim gereği duyabileceği her birime bir "port" numarası verilir. (Bu numarayı kapı numarası veya telefon numarası gibi de düşünebiliriz). Klavye, disk(ler), ekran, programlanabilen saat, vs gibi birimlerin, farklı port numaraları vardır. CPU, bu birimlerin her hangi biriyle veri alış verişini yapmak istediğinde o birime ilgili port numarasını göndererek yapar. 8086 CPU en çok 65,336 farklı port numarasına erişebilir. Bu numaraların hepsi kullanılmaz (çünkü gerekmez). Hangi port numarasının hangi birime ait olacağını bilgisayar tasarımcısı saptar.

"Port"ların hemen tamamı sadece en alt seviye işletim programı olan BIOS tarafından kullanılır.

Bilgisayarın dışındaki çevre birimlerinin, (örnek:yazıcı, modem) bilgisayara bağlantıları seri ve/veya paralel portlarla olur. Bu portlar donanım arabirimi yani konnektörler ve veri gönderme/alma yollarıdır. CPU, bu portlara dahi kendi özel numaralarıyla ulaşır.

Registers (Yazmaçlar veya geçici kayıt birimleri)

Bir register CPU'nun içinde bulunan, komut veya verilerin geçici olarak kaydedildiği, özel amaçlı küçük bir bellektir. (CPU tipine göre 16 veya 32 bit genişliğinde.)

CPU içinde register'ler bir kaç gruba ayrılmışlardır. 386 öncesi CPU'larda 14 register programcının kullanımına veya kontrolüne ayrılmıştır. 386 ve sonrası CPU'larda bu sayı 16 dir.

Diğer register'lar CPU'nun kendi kullanımı içindir. 386 ve sonraki CPU'larda bu registerlerin sayısı 32 dir. 386 öncesi CPU'larda 4 adet genel amaçlı 16 bit'lik register bulunur. Bunlar AX, BX, CX ve DX tir. 386 ve sonraki CPU'larda ise 8 adet 32 bit'lik genel amaçlı register bulunur. Bu registerler: EAX, EBX, ;ECX, EDX, ESI, EDI, EBP, ESP dir.

Programcı genel amaçlı registerları geçici kayıt alanı ve hesaplamalar için yaz/sil olarak kullanabilir.

Programcı kullandığı yöntemin gereği isterse yukarıda adı geçen registerlerin sadece yarısını kullanabilir. Örneğin AX registeri AL ve AH olmak üzere iki 8 bit'lik dilimlere (slices) bölünmüştür. Aynı olanak 32 bit'lik registerler için de geçerlidir ve bu registerler da 16 bit'lik kısımlar halinde kullanılabilir.

Diğer bir registerler kümesi ise "Segment" (Kısım) registerleridir. Belleğin farklı kısımlarına (veya dilimlerine) erişmek için kullanılırlar. 386 öncesi CPU'larda 4 adet olan Segment register sayısı, 386 sonrası ve Pentium dahil altı adettir.

386 öncesi CPU'lar belleğe 64K'lık kısımlar halinde erişebilirken, 386 ve sonraki CPU'lar (korunmalı konumda) 1 byte ve 4Gbyte arası değişebilen (programcının tercihine göre) bellek kısımlarına (erişebilme olanağına) sahiptir.

CS-Code Segment bellekte programın bulunduğu kısmı gösterir.

DS-Data Segment programın kullandığı verilerin yerini gösterir.

ES-Extra Segment ise veri kısmı registerına (gereklikçe) ek olarak kullanılır.

SS-Stack Segment (yığın kısmı) ise birazdan incelenecek olan bellek diliminin yerini belirler.

386 ve sonraki CPU'larda bulunan FS ve GS registerleri ise programcılara, deęişik bellek (yazılım) düzenekleri arasından seçim yapabilme olanađını verir.

Bir başka registerler kümesi ise, SS registeriyle birlikte kullanılarak, bellekte belli (özellikle istenilen) byte'lara ulaşmak için kullanılır. Bunlardan:

IP-Instruction Pointer (Komut göstericisi veya işaretleyicisi) işlemcinin işleyeceği bir sonraki program komutunun yerini (adresini= CS adresi + IP adresi) tutar.

SP-Stack Pointer (yığın göstericisi) ve.

BP- Base pointer (taban göstericisi) ise Stack üzerine (içine) kaydedilmiş olan "iş" in ilerlemesini izlemeye (ve kontroluna) yardımcı olurlar.

Stack (yığın) bellek üzerinde, CPU'nun o esnada yaptığı "iş" hakkında kayıtların tutulduğu bir bellek alanıdır. Bir bilgisayar çalışması süresinde, giderek artan karmaşık işler "yığın"ıyla karşı karşıya kalır. Bu durumda bilgisayarın (CPU'nun) sürekli olarak nerede olduğunu ve ne yaptığını bilmesi gerekir. CPU yaptığı "iş" in bir yerinden diğerine gidip gelirken, yine bir önceki kısma dönebilmek için nerede ve nasıl bıraktığını bilmesi gerekir. CPU her yeni "iş" e (task) başlarken ,bir önceki "iş"le ilgili bilgiler "yığın"ın üstüne itilir (push). En son "iş" (task=görev) bittiğinde ise bu bilgiler yığından çıkarılır ve dahada önceki "iş" le ilgili kayıtlar yığının geçerli kayıtlarını oluşturur.

SI-Source Index (kaynak endeksi) ve

DI- Destination index ("varılacak yer" endeksi) registerleri ise programların, büyük miktarda verileri bellekte bir yerden diğer bir yere aktarmasına yardımcı olur.

Flag Register ("Bayrak" yazmacı)

"Bayrak" işaretleme flaması anlamının bilgisayar terminolojisine yansıtılmasıdır. CPU'nun çeşitli durum ve konumlarını belirten birbirinden bağımsız 1 bit'lik bilgilerin (Flag) bir arada (fakat birbirlerinden ayrık olarak) bulundurulduğu registerdir.

386 öncesi CPU'larda 9 standart "flag" bulunur. 386 ve daha sonraki CPU'larda ise 32 bit'lik EFLAGS registeri, CPU cinsine göre 13-17 "flag" bulunur. Pentium'da 17 "bayrak" bulunur.

Bu bayraklardan 6 sı aritmetik ve benzeri işlemlerin sonucunu belirtir. Bunlardan: ZF-Zero flag (sıfır bayrağı) sonucun 0 olduğunu (eşitlik karşılaştırması),

SF-Sign fiag (işaret bayrağı) sonucun eksi olduğunu,

F-Carry flag ("elde" bayrağı) bir sonraki basamağa taşınması gereken bit'in varlığını

AF-Auxiliary carry flag (yardımcı "elde" bayrağı) ilk dört bit'ten gelen "elde" olup olmadığını,

OF-Overflow flag ("taşma" bayrağı) sonucun kayıt alanına sığamayacak kadar büyük olduğunu,

PF-Parity flag (parite bayrağı) sonucun tek veya çift paritesini gösterir.

Diğer 3 bayrak, kontrol amaçlı kullanılır,

DF-Direction flag (yön bayrağı), tekrarlanan işlemlerde (byte, byte veri kaydırılması gibi) sola veya sağa yön belirtir

IF- Interrupt flag, ("kesme" bayrağı) "Kesme" işlemlerinin yapılıp yapılamayacağını gösterir

TF- Trap flag ("Kapan" bayrağı) CPU'nun bir komutu işledikten sonra, bir "kapan" "kesme" işareti üreterek, durmasını sağlar. Bu durum programın belli bir adımdaki sonuçların izlenebilmesini sağlar. Bu bayrak bilhassa program hatalarını gidermede önemli bir olanaktır.

286 CPU'da yukarıdakilere ek olarak iki bayrak daha ilave edilmiştir: NT-Nested task flag ("yuvalanmış", biri diğerinin içinde bulunan)

"işler" (tasks) bayrağı, ve

IOPL-I/O Privelege Level flag (2 bit'lik giriş çıkış öncelik seviyesi bayrağı).

386 CPU'da yine iki ek bayrak daha vardır.

RF-Resume flag ("Devam et" bayrağı) program hatalarını gidermekte kullanılır.

VM-Virtual mode flag (Sanal kip bayrağı) İşlemciyi sanal 8086 konumuna geçirir.

486 CPU'da sadece 1 ilave bayrak daha vardır.

AC-Alignment Check ("Uyumluluk" kontrolü bayrağı) bellek adres referanslarının belli sınırlara uyup uymadığını gösterir.

Pentium işlemcisi ise tümüne ilaveten 3 yeni bayrak daha getirmiştir.

VIF- Virtual interrupt flag (Sanal "kesme" bayrağı). Daha önce yazılım olarak gerçekleştirilen bu bayrak, çok görevli (multitasking) ortamlarda bazen problem yaratan IF bayrağını kullanmamak için uygulanmıştır.

VIP- Virtual interrupt pending flag (Sanal "kesme" hazırlık bayrağı) aynı amaçlara yardımcıdır.

ID-Identity flag (Kimlik bayrağı) programlara, her hangi bir uygulamanın ne tip işlemci sınıfında çalıştığını belirler. (İşlemci kendini ve hangi kipte olduğunu belirtir. [gerçek, korumalı veya sanal 86 kipleri])

Bölümlenmeli adresleme yapısı (Segmented)

Bu adresleme yapısında bellek alanı 64KB yer kaplayan parçalara ayrılır. Her bölüm 16'ya bölünelilen bir sayıdan ibaret olan bir paragraf adresinden başlar. Her byte veya word'e ulaşmak için o bölümün içinde belli bir yeri gösteren bir offset değeri kullanılır.

Paragraf ve offset adresi bir araya geldiğinde tam bir adres oluştururlar. İşlemci oluşan bu 32 bit adresi 20-bit gerçek adrese indirger. Örneğin adresimiz 1234:4321 ise aşağıda görülen toplama işlemi sonucunda 16661 fiziksel adresi oluşur.

```
12340
+ 4321
-----
16661
```

Bu adresleme şekli PC tabanlı sistemlerin tüm bellek adreslemesinin temelidir.

2.2.9 Kesmeler

Kesme işlemciye ani bir isteğin bildirilmesinde kullanılan bir haberleşme yöntemidir. 80x86 temelli işlemciler kesmeleri donanımdan veya yazılımdan alabilirler. Bir donanım cihazı PIC (programlanabilen kesme kontrolörü) tarafından işlenebilen kesme üretim yeteneğine sahiptir. Ardından bu kesme isteği işlemciye bildirilir. Yazılım tarafında ise INT komutu bir kesme üretilmesini sağlar. Her iki durumda da işlemci o anda yaptığı işi durdurarak bellekte sabit olarak durmakta olan bir altprogram olan kesme kotası programı çağırır. Bu altprogramın işi bittiği anda işlemci kaldığı noktadan işe devam eder.

8086 256 farklı kesmeyi destekler. Bu 256 kesme kotası programların bulunduğu yeri gösteren adres tablosu 0000:0000 adresinden itibaren yerleşmiştir. Her kesme vektörü 4 byte boyundadır..

PIC çevre birimlerden gelen istekleri mikroişlemciye sırayla bildiren bir çiptir. PIC'ten (Programmable Interrupt controller) gelen her sinyalle birlikte mikroişlemci o anda yapmakta olduğu işi bırakarak -eğer uygunsa-, gelen sinyalle ilgili kodu çalıştırır; işlem bittiğinde ise yarım bıraktığı koda tekrar kaldığı yerden devam eder. Bu yolla mikroişlemcinin zamanı rasgele oluşan sinyalleri izlemekle harcanmaz, sadece sinyal olduğu anda mikroişlemcinin ilgili işi yapması sağlanır. Basılan her tuş, seri kanaldan gelen veya giden bilgiler, paralel kanaldan gelen veya giden bilgiler, VGA kartına giden veya karttan gelen bilgiler, DMA çipinin işini bitirmesi, Timer'ın ayarlanan zamanının dolması birer kesme kaynağı olabilirler.

PIC çeşitli kesme isteklerine öncelikler atayabilir. Örneğin, PC'lerde en yüksek önceliğe sahip kesme, IRQ0 bacağında sinyal veren ve 8. Kesme olarak atanmış olan timer tick kesmesidir. Eğer timer kesmesi işlenirken daha düşük öncelikli bir başka kesme gelirse, bu kesme timer işini bitirinceye kadar bekletilir.

Yalnız tek bir donanım kesmesi tüm PIC'i atlayarak işlemciye ulaşır. Bu kesme NMI'dır. (Non-maskable interrupt - Maskelenemez kesme). Özel olarak bir bellek donanımı hatası olduğunda bu kesme çalıştırılır.

Timer belli bir zamana programlanarak zaman dolduğunda bir interrupt üretme görevini üstlenir. Genelde bir kanalı sabit olarak programlanır ve belli aralıklarla yapılması gereken işlerin başlatılması için interrupt üretir. Bu işler dinamik belleklerin tazelenmesi, bazı kontrollerin yapılması ve kilitlemelerin kontrolü tipi işlerdir.

DMA bellekten belleğe veya bellekten I/O portlarına, mikroişlemcinin zamanını çalmadan bilgi aktarmak için kullanılır.

2.2.10 Çevre kontrol üniteleri

Çevre ünitelerin bazıları kasa içinde bazıları ise kasanın dışında bulunur.

Kasa içinde bulunan temel çevre üniteler VGA controller kartı, multi I/O kartı ve bu karta bağlı olan ünitelerdir.

Multi I/O kartı üzerinde bir floppy controller, bir hard disk controller, iki seri port controller ve bir game controller bulunur. Floppy controller maximum iki adet floppy ünitesini kontrol eden bir birimdir. Floppy ünitesi küçük kapasiteli bilgi bir depolama aletidir. Hard disk controller maximum iki adet sabit diski kontrol eden bir birimdir. Hard disk ünitesi orta hızlı, yüksek kapasiteli, enerjisi kesilse bile bilgileri saklamaya devam eden bir ünite dir.

Seri portlar temelde iki ayrı interrupt ve dört ayrı adres kombinasyonlarından birine yerleştirilebilirler. Seri portlar diğer bilgisayar temelli aletlerle haberleşmek için kullanılan standart bir arabirimdir. Seri portlara -genelde ana kasa dışından olmak üzere- modem adlı alet bağlanır. Modem seri kanaldaki bilginin telefon hatlarıyla aktarılmasını veya telefon hatlarından gelen bilginin seri port tarafından alınmasını sağlayan bir alettir. Seri port üzerinde standart bir asenkron haberleşme yürütülmektedir.

Game controller ise oyun oynarken kullanılan joystick adlı aletin bağlanmasını sağlayan porttur.

Dışarıdan takılan birimlerin en önemlileri ise tuş takımı, VGA monitor ve modemdir. Tuş takımı kullanıcı arayüzünün bir parçası olup kullanıcının girdiği bilgilerin bilgisayar tarafından anlaşılmasını sağlar. Kullanıcının bastığı her tuş bir byte bilgisine çevrilerek keyboard controller çipine yöneltilir. Keyboard controller bu byte'ları okuyarak bir kesmenin çalışmasını sağlar. Bu kesme sonucunda , ortaya, anlamı basılan tuşun ASCII karşılığı olacak şekilde bir başka byte daha çıkar ve bu bilgiler keyboard buffer'a her basılan tuşa iki byte karşılık düşecek şekilde yazılır.

VGA monitor ise kullanıcı arayüzünün bir başka parçasıdır ve bilgisayarın kullanıcıya aktarmak istediği bilgilerin kullanıcı tarafından anlaşılmasını sağlar.

VGA hem grafik hem de karakter görüntüyü destekleyen bir sistemdir. 80 kolon 25 satır karakter modunda sekiz ayrı görüntü sayfası olabilir ve bu sayfalardan isteneni aktif yapılabilir. Programlar genelde ilk dört sayfayı kullanırlar. Her sayfada 80*25*2 karakter vardır. Çift numaralı byte'larda karakter bilgisi, tek numaralı byte'larda ise renk kodu tutulur.

Modem uzak noktadaki diğer bir bilgisayar veya benzer makinayla haberleşmede kullanılır. Genelde uzak noktadaki bilgisayar üzerinde, yakın noktadaki kullanıcının girdiği verileri değerlendirip kendisine cevap veren bir program olur. Bu programın olmadığı durumlarda ise kullanıcı standart bir terminal emulasyon programı ve modemi kullanarak bir başka kullanıcının ekranına mesaj atabilir ve ondan mesaj alabilir.

PC'de "ekranın" canlı kısmı fiziksel ekranın (resim tüpü yüzeyinin) tamamını kapsamaz. Ekranın her iki yanı, alt ve üst kısımlarında boşluklar vardır. Elektron demeti buraları da tarar ancak sınır bölgelerinde bilgi gösterilemez. (Bu kısımlardaki "tarama" parlaklık kontrolü sonuna kadar açılarak görülebilir.) Normal olarak "siyah" olan "sınır" (border) bölgelerinde bilgi gösterilememekle birlikte, "siyah" tan başka bir renkle de gösterilebilir. Böylelikle sınır ve 'arka plan" renkleri uyumlu olursa göz daha az yorulur.

PC'ler başlangıçta karakter (yazı) ağırlıklı kullanımlar için tasarlanmış, gösterim standartları da buna göre geliştirilmişti. Ancak zaman içinde Grafik ekranların getirdiği işletim kolaylıkları dolayısıyla daha fazla tercih edilmeleri, PC'lerde de grafik ekran konum ve standartları geliştirildi. Bugün her PC'de her iki "gösterim" konumu da, kullanıcının tercihine göre uygulanabilmektedir.

Grafik konumunda "resim elemanları" (picture elements), kısaca "piksel" denilen küçük noktacıklar, resmi oluştururlar. Piksel sayısı belli bir resimde elde edilebilecek en küçük ayrıntıyı belirler.(Resolution=Seçicilik [çözünürlük]). Bugün en yaygın kullanılan grafik çözünürlüğü 640*480 ve 600*800 çözünürlükleridir.

Kişisel bilgisayarların çok büyük çoğunluğu 25x20 cm görüntü alanı içeren "37 cm" ekranlardan oluşur. Bu boyutta 640x 480 piksel üzerinde seçicilik (çözünürlük) kullanınca,"ekran" alanı üzerine daha çok "detay" mümkün olmakla birlikte görüntü boyutları da orantılı olarak küçülmektedir. Yaygın, şekilde kullanılan Word ve Excel (ve birçok diğer) gibi programlarda 640x480 en "uygun" (optimum) görüntüyü vermektedir. Metin modunda her ekran bilgisi iki byte'tan oluşur.

İlk byte PC karakter setindeki, 256 (2^8) farklı karakterden biridir. Bu karakterlerden bazıları ekranda görülmez. Satır başı, bir sonraki satıra atlama, harfler veya kelimeler arasındaki boşluk işaretleri, görünmeyen işaretlerin tipik örnekleridir.

2. Byte ise üç kısımdan oluşur. İlk 3 bit ön plan rengini (karakterin rengini) belirtir. Bu şekilde 8 farklı renk elde edilir. 4. bit parlaklık bitidir. Bu bit "0" ise ilk 3 bit'in belirttiği S renk olduğu gibi nispeten daha koyu görünür. 4. bit "1" olduğunda önceki "parlaklığı" daha fazlası saklanarak ilk sekiz rengin ikinci kombinasyonu elde edilerek toplam 16 renk oluşturulur.

0000: Siyah veya kahve)	0110: Yeşil+Kırmızı (koyu sarı
0001: Mavi (=Beyaz)	0111: Yeşil+Kırmızı+Mavi
0010: Yeşil	1000: Daha parlak (Gri)
0011: Mavi+Yeşil" (Cyan) (parlaklık+Mavi)	1001: "Açık' mavi
0100: Kırmızı	1010: "Açık' yeşil
0101: Mavi+Kırmızı (Magenta)	1001: "Açık' mavi+yeşil (Cyan)
1100: 'Açık" kırmızı	1110: "Açık" sarı
1101: "Açık" Magenta	1111: "Parlak' Beyaz.

Bu tablodan anlaşılacağı üzere daha çok "renk" için daha çok bit gerekmektedir. 2. Byte'ın 5,6,7. bitler; yine Mavi, Yeşil ve kırmızı kombinasyonlarıyla, 8 farklı arka plan (zemin) rengini verir. Arka planda parlaklık bit'i olmadığı için 'açık" renkler yoktur.

Karakter konumlarının bir diğer ögesi ise imleç'tir (Cursor). İmleç ekranın faal yerini belirtir. Kullanıcıya, yazılacak bir sonraki karakterin yerini belirtir. İmleç karakter konumları için (genellikle) "yanıp sönen" alt çizgidir, ve video kartı devreleri tarafından üretilir.

Seri Port (RS-232 Port)

İsminden de anlaşıldığı gibi seri port'ta bir hat üzerinden veri bitlerini ard arda göndererek (TX), diğer bir hattan da gelen bitleri ard arda alarak (RX) veri transferi yapılır.

Seri porta bağlı kablonun uzunluğu 15-20 mt arası olabilir. Seri port bilgisayarlar arası bağlantıda kullanılmakla birlikte, yazıcı ve seri protokolla iletişim yapabilen diğer herhangi bir birime de bağlanabilir.

Seri port bilgisayar kasasının arka tarafında 9 veya 25 bacaklı (bazen her ikisi de) erkek konnektör(ler) olarak bulunur.

Transmit Data (TX) ve Receive Data (RX): Veri gönderme ve alma hatları.

Carrier Detect (CD veya DCD): Modem'lerin birbirleriyle haberleşmesinde kullanılır. Karşı modemden gelen "taşıma" (carrier) işaretini gören yakın modem, bu hat üzerine pozitif bir işaret bindirerek, bilgisayara, karşı modemle iletişimin kurulduğunu belirtir.

Data Terminal Ready (DTR): Bilgisayardan modeme gönderilir, bilgisayarın "hazır" olduğunu bildiren bir pozitif işaretir.

Data Set Ready (DSR): Modemden bilgisayara DTR işaretine karşı gönderilen (pozitif "hazır" yanıtı). DTR/DSR protokolunun tamamlanmasından sonra bilgisayar ve yakın modem arasında iletişim başlamaya hazırdır. Ancak aşağıdaki iki işaretin de tamam olması gerekir.

Request To Send (RTS): Bilgisayarın, modeme veri transferine hazır olduğunu ve modemden onay isteyen pozitif işaret.

Clear To Send (CTS): Modem RTS işaretinin onayını, pozitif CTS işareti ile verir.

Bu durumda, bilgisayar ve modem arasında iletişim başlayabilir.

Ring Indication (RI): (Zil belirtisi) Modemden Bilgisayara gönderilir ve karşı modemden bir çağrı geldiğini belirtir.

Modem cihazı, telefon hatları üzerinden uzak bilgisayarlara iletişim kurmak için kullanılır. Modem cihazının iletişim hızına "Baud" denilir. Eski bir tanım olan Baud yerine bugün daha çok saniyede gönderilen bit sayısı kullanılmaktadır.(Örneğin 300 Baud yaklaşık 300 bit/saniye demektir)

Telefon hatları 300-3400 Hz arası ses işaretlerini iletebilirler. Diğer bir deyişle bir saniyede en çok 340 bit gönderilebilir gibi gözükmektedir.

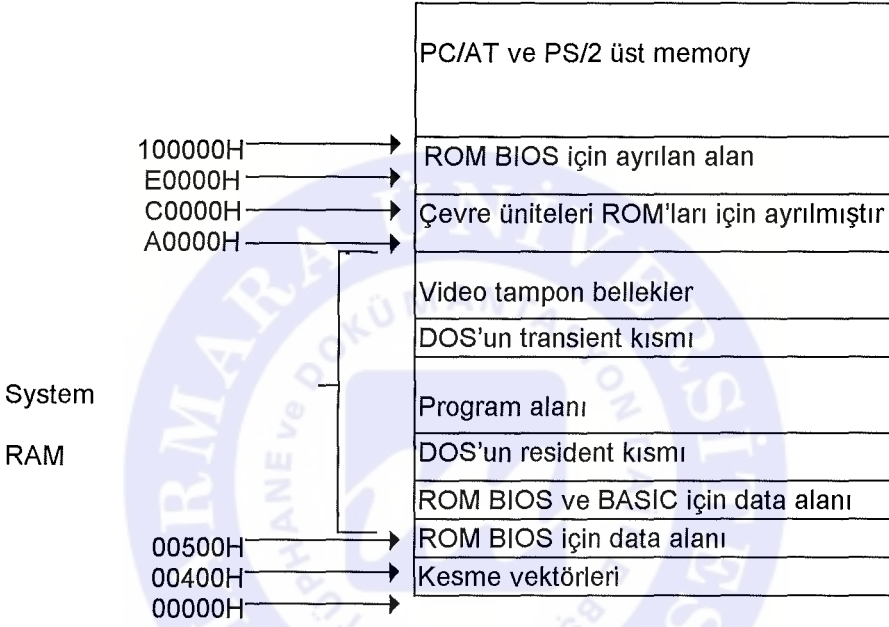
Buna rağmen günümüzde modem cihazları değişik kodlama teknikleri kullanarak en az 14400 baud rate'de veri gönderip alabilirler. Daha gelişmiş modellerde bu hız daha yüksektir.

Tümleştirme teknolojisindeki gelişmeler sonucu modem cihazları, günümüzde artık harici bir birim olmaktan çıkmış ve bilgisayarın içine takılan ve ayrıca fax işlevini de içeren kartlar haline dönüşmüştür. Telefon hattı artık doğrudan; bu kartın arkasında bulunan telefon hat bağlantısına birleştirilmektedir.



2.3 PC YAZILIM YAPISI

Önceki sayfalarda anlatılan PC donanımı oldukça uzman yazılımlar tarafından yönetilirler. Tüm temel girdi/çıkı işlemleri BIOS'un içine kodlanmıştır. BIOS çalışmasının ardından işletim sistemini aramaya başlar. İşletim sistemi yüklendiğinde üzerine değişik programlar yüklenebilir. Genelde bu programlar girdi/çıkı işlemleri için işletim sistemine istek bildirirler. İşletim sistemi de bu isteği gerçekleştirirken ya BIOS'u kullanırlar -DOS gibi- ya da kendi kernel'larını kullanırlar-UNIX, WINDOWS gibi-



Şekil 2. PC bellek kullanımı

2.3.1 KOMUT SETİ

Makina Dili (Machine Language)

Makina dili, kısaca, bilgisayar komutlarının 2 veya 16 sayı tabanına göre ifade şeklidir. 0 halde komutlar:11010011 veya D3 gibi ifade ediliyorsa, komutlar makina dilindedir. Bu şekilde, bilgisayarlar en alt düzeyde programlanabilir. Ancak bu şekilde programlama uzun süreli ve yorucudur.

Assembler Dili (Assembly Language)

Esas itibariyle Assembler ; "Toparlayıcı" veya "Bir araya getirici" anlamına gelir. Makina dilinin insanların daha kolay anlayacağı şekilde harflerle (kısaltılmış kelimelerle) ifade edilmiş birimdir.

Buradaki önemli nokta, Assembler dilinde program yazan bir kişi, bilgisayara yaptırmak istediği işleri en küçük ayrıntısına kadar bir, bir yazmak zorundadır. Bu durumun aksine "Yüksek Seviye Programlama Dillerinde" programcı daha kolay ve basit yöntemlerle istediği işlemleri yaptırabilir. Ancak burada yazılan program ki buna kaynak program kodu (source code) denir, bir başka programın (derleyici- compiler) yardımıyla makina diline çevrilir.

Assembler Dilinde Komut Kümesi

Assembler dilinde MOV DX, CS komutu, CS registerının içindeki verileri (bitleri) AX registerine taşı (veya naklet) anlamına gelir. MOV, İngilizce Move (kaydır, naklet, taşı hareket ettir.) kelimesinin kısaltılmış şeklidir.

Benzer şekilde: MOV AL, BL komutu; BX registerının alt byte'i olan BL kısmının, AX registerının alt byte'i olan AL kısmına taşı anlamına gelir. Diğer bir örnek: MOV AL,11 komutu AL register kısmına doğrudan 11 (verisini) taşır (nakleder). Daha başka bir örnek: MOV CX, [BETA] komutu BETA bellek adresinin içeriğini CX registerine naklet anlamına gelir.

MOV komutu ile herhangi bir register'dan diğer bir register'e (aynı alan genişliğini kullanmak şartıyla); Herhangi bir registerden herhangi bir bellek adresine, ve herhangi bir bellek adresinden herhangi bir register'a veri taşımak olanağı vardır.

MOV komutuyla taşınan veri, kaynağında aynen kalır, yani varacağı yere kopyalanır. Varacağı yerdeki bir önceki veri silinir. Bazı hallerde ise bir önceki verinin korunması gerekir. Bu gibi hallerde XCHG (exchange-yer değiştir) komutu kullanılarak iki ayrı register arası veya bir register ile bellek adresi arasında yapılabilir. Örnek: XCHG AX, DX

PUSH komutu, bir registerin içeriğini yığın (stack) üzerine "koymaya" yarar. Örnek PUSH AX.

POP komutu ile yığın üzerine konmuş olan register bilgisi (içeriği) geri alınır.

PUSF, POPF komutları ile "bayrak" registeri bilgileri yığın üzerine konur ve geri alınır.

IN, OUT komutları ise portlardan bilgi almaya ve göndermeye yarar.

Diğer bir komut alt kümesi ise toplama işlemleri yapar. Bunlardan bazıları: ADD, ADC, INC. dir. ADD AX, BX komutu AX ve BX register içeriklerinin toplanarak toplamın AX registerine yazılmasını sağlar. ADC komutu "elde" sayısının işleme girmesini sağlar.

"Elde var" sayısı "elde" bayrağından alınır. INC (Increment-arttır) komutu belirtilen register veya bellek adresine 1 ilave eder.

Benzer şekilde çıkartma işlemleri yapan komutlar mevcuttur. Bunlardan bazıları: Toplama komutlarına karşılık gelen SUB, SBB, DEC' dir. DEC (Decrement-Azalt) komutu belirtilen register'i veya bellek adresini 1 azaltır.

MUL, DIV komutları da çarpma ve bölme yapmaya yarar.

AND, OR, NOT komutları mantık işlemleri yapmak için kullanılır.

CMP: komutu (Compare-Karşılaştır) iki registerin içeriğini, veya bir register ve bir bellek adresinin içeriğini, veya bir registerin belli bir sayıyla veya bir bellek adresinin belli bir sayıyla karşılaştırır. Karşılaştırmanın sonucu çeşitli "bayrakların" konumunu etkiler.

JMP komutu (Jump-Atla) programın işlem sırasını değiştirir. Bilgisayarın "karar verme" yeteneğinin sırrı da budur. JMP komutunun argümanı "bayrak" registerindeki farklı bitlerin durumudur. JMP tek başına da kullanılabilir. Bu duruma koşulsuz atlama denilir. Örneğin JMP BX komutu, program işlem sırasının BX registerinin içeriğinin gösterdiği bellek adresine atlar. Koşullu JMP örneği ise şu şekilde olabilir. JZ BX komutu yukandaki gibi ancak sıfır bayrağı (zero flag)=1 ise işlem görür.

Yukarıda verilen komut örnekleri 8086 işlemcinin komut kümesinden alınmıştır. 8086'nin komut kümesinde 89 farklı komut bulunur. Bu komutların ayrıntılı incelenmesi ise konumuzun dışındadır.

Burada vurgulanmak istenen bilgi, yukarıda ayrıntıları verilen komutları (daha doğrusu bu komutlara karşılığı olan ikisel tabanlı sayıları) işlemci (CPU) doğrudan tanır ve gerekli elektronik işlemleri adım adım gerçekleştirir.

286 CPU komut kümesi ise, 8086 komutlarının tamamını tanır ve ilave komutlar içerir. 286 işlemcinin toplam 113 komutu vardır. Böylece 286, hem 8086 için yazılmış olan programları, hemde yeni komutların sağladığı olanakları kullanan yeni programları çalıştırabilir (koşturabilir).

386 CPU komut kümesi de yine 286 komut kümesinin üstüne yapılandırılmış olup 150 den fazla komutu içerir.

486 CPU komut kümesi 386 ve 387 komut kümelerini artı başka komutları da içerir. Bu komutların yapısı ve ayrıntıları daha çok sistem ve assembler dilinde programlama yapan kişileri ilgilendirir.

2.3.2 BIOS

Programlar en verimli şekilde çalışabilmeleri için "katmanlar" (layers) halinde çalışacak şekilde tasarlanırlar. BIOS ise fiziksel katmanın üzerindeki en alt katmanı oluşturur. BIOS ana işlev olarak donanımın en temel gereksinimlerine (yönetim ve denetim) olanak sağlar. BIOS ROM'u tüm altseviye kontrolleri gerçekleştiren bir yazılımı içinde barındırır. BIOS tüm tek kullanıcıli işletim sistemlerine destek verebilecek yetenekte bir standart sunar. Bu nedenle DOS gibi işletim sistemleri genelde BIOS'u kullanmadan donanıma direk ulaşmak yerine BIOS üzerinden ulaşırlar. Herhangi bir sistemin donanımı diğerlerinden olukça farklı bile olsa BIOS kismelerini desteklemek zorundadır. Bu nedenle BIOS kismelerini kullanarak yazılan her program benzer işleri değişik şekillerde yapan tüm donanımlara destek verecek durumdadır.

BIOS ilk açılışta

1. PC'nin kendi kendini kontrol etmek (Power on self test),
2. Her donanım cihazına ilk değerlerini atamak (initialisation). (Başlatma işlemlerinde her bir dahili donanım biriminin çalışmaya ilk başlarken gereksinim duyduğu değerleri bu birimlere gönderir).

3. Kendi içinde kayıtlı donanımın var olup olmadığını kontrol etmek ki bunu yeni model PC'lerde kısmen standart donanımın kayıtlı olduğu bir başka sürekli (silinmeyen) belleği okuyarak yapar,

4. "Kesme" (Interrupt) adreslerini RAM'e yazmak, ve son olarak ta

5. Önce disket sürücüsünü ve sonra diski deneyerek işletim sistemini yüklemeye çalışmak gibi işlemleri yapar.

BIOS işlemlerinin sonuçlarından bazıları, örneğin RAM test, özel donanım tanımı (video kart vs.), CPU ve yardımcı işlemci tanımı, disket ve disk tanımı; portların tanımı başlangıçta ekranda görülebilir.

İkinci olarak, BIOS normal çalışma süresi içinde, donanım "kesmelerini" ve ilgili "kesme" programını (interrupt routine) devreye sokar.

Üçüncü kısımda ise diğer programların hizmet taleplerine cevap verebilecek şekilde beklemeye başlar. Bir başka program BIOS'tan iş isteğinde bulunuyorsa ilgili kesmeyi kullanır. Bu kesmelerin listesi aşağıdadır.

Kesme Numarası (Hex)

05	Ekranı Yazdır (Print Screen)
1C	Video (Ekran gösterimi)
12	Bellek Tespiti
13	Disket ve Disk
14	Seri Port iletişimi
15	Muhtelif Sistem Hizmetleri
16	Klavye
17	Yazıcı (Paralel Port)
18	Ağ (Network) arabirim kartı
19	ROM-BIOS'un kendi kendini "gölge ram" (Shadow Ram) sahasına yüklemesi
1A	Sistem zamanlayıcı, Gerçek zaman saati
4b	İleri seviye hizmetleri (SCSI, DMA)

Yukarıda görülen hizmet (service) gruplarına, her birisi için özgün kesme numarası programlanarak ulaşılır. Bu kesmelerin bir alt grubu da hizmet numarasıdır. Bu hizmet numaralarını çok ayrıntıya girmemek için burada vermiyorum. Ama ilgilenen kişiler IBM BIOS Technical Manual adlı kitaptan veya ekte verdiğim kaynaklardan yararlanarak bulabilirler.

2.3.3 İŞLETİM SİSTEMİ

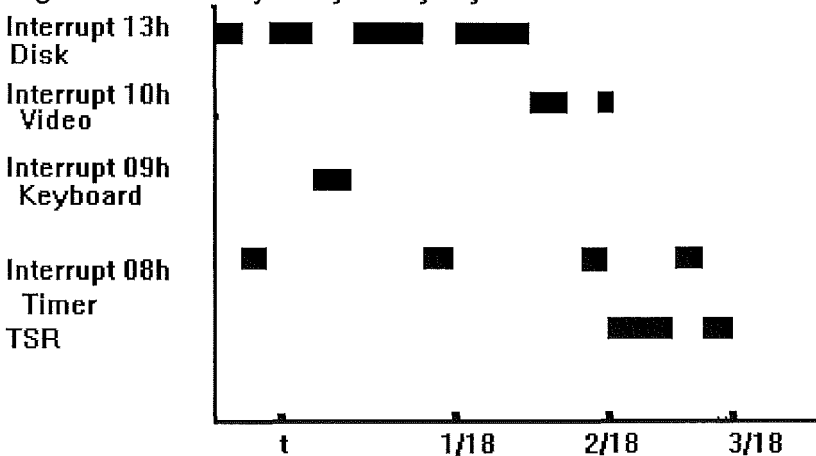
İşletim sistemi uygulama programlarıyla sistem donanımı arasında bir katmana yerleşen bir yazılımdır. Bu katmanın yeri sabit değildir çünkü işletim sistemi bazı durumlarda donanıma doğrudan hükmeder, bazı durumlarda ise ara katmanlar üzerinden -BIOS, KERNEL gibi- bağlanır. İşletim sisteminin uygulama programlarına en yakın olan kısmı device driver denen, programların sistemdeki birimlere ulaşmasını kontrol eden bölümdür. Device driver'lar o işletim sisteminin yapısına uygun olarak kendinden beklenen işleri yerine getirir veya karşı çıkarlar. Örneğin DOS'ta hiç bir kontrol yokken bir dosyayı rahatça her kullanıcı açabilir ve her noktasına ulaşabilir. Device driver'ın özellikleri kullanılarak o dosyanın bir bölümünün veya tümünün diğer kullanıcılara kilitlenmesi sağlanabilir. UNIX gibi çok kullanıcılı işletim sistemlerinde ise Device Driver'lar dosyanın bazı kullanıcılara kilitlenmesini sağlayabilirler.

2.3.4 KESME YÖNETİMİ

DOS işletim sistemi altında sistemde iki tür kesme oluşur. Bunlar donanım ve yazılım kesmeleridir.

Donanım kesmeleri sistemin içinde bulunan çevre ünitelerden gelen sinyaller sonucu oluşur. Bu kesmelerin hepsi veya sadece bazıları interrupt controller çipi yardımıyla engellenebilir. Bu kesmeler normalde BIOS ve DOS tarafından yakalanır ve ilgili kodun çalışması sağlanır.

Yazılım kesmeleri bir program içinde bir başka alt programı çağırmak gibidir. Bu kesmelerin kodu BIOS içine yazılmıştır. Bu kesmeler sistemlerin birbirinden farklı oluşunu gözdardı etmek için kullanılır. Birbirinden donanım olarak farklı olan sistemlerde BIOS'u kullanan programlar donanımdan bağımsız olarak uyum içinde çalışırlar.



Şekil 3

Her iki tür kesme de yazılan bir koda yönlendirilebilir. Böylece PC-DOS sisteminde her tuşa basıldığında, seri port'dan her karakter geldiğinde özel bir iş yapmak mümkün olur.

2.3.5 RESIDENT PROGRAMLAR(TSR)

Bu katman yapısı anlatılan şekilde sabit olarak kurulmasına rağmen, donanım bilgisine sahip olan yazılımcılar tarafından çeşitli noktalarından delinebilmekte ve işletim sistemini, BIOS'u veya kernel'i değiştirmeden boyutu son derece küçük olan programlarla sistemde çok büyük değişiklikler yapılabilmektedir. Bu tür programlar genelde device driver olarak veya resident programlama tekniği kullanılarak yazılır.

Genelde sistemdeki her hardware device'a karşılık bir software device driver vardır. Device driver ilgili hardware'i kullanmak için gerekli I/O kodlarını içerir. Device Driver paylaşımın düzenlenmesini sağlar, aynı device'ı birden çok kişi kullanmaya kalkıştığında onları engeller veya kuyruğa sokar.

Resident programlama tekniğinde donanım kesmesi direk resident programa yönlendirilerek kesmenin bu program içinde değerlendirilmesi sağlanır. Böylece gelen sinyalin işletim sistemine bile ulaşmaması sağlanabilir. Örneğin işletim sisteminin sinyali farketmeden basılan her tuş bilgisi bir dosyaya kaydedilebilir veya seri porttan atılabilir.

Seri port'a erişecek bir programcı ya doğrudan BIOS'u ya da işletim sistemindeki device driver'ı kullanır. BIOS'u kullanması durumunda BIOS'u standartlaştırmak adına üzerine yüklenen tüm hantallığa katlanmak zorunda kalır. İşletim sisteminin DOS olması durumunda ise Device driver'ın oldukça yetersiz özelliklerle donatıldığını görür. Bu nedenle bir programcı seri port'u kullanması gerektiğinde seri port'a doğrudan erişmeyi tercih eder. Bu durum incelediğim bir çok yaygın terminal emülasyon programında böyledir. BIOS'u kullanmak standart dışı makinalarda da programların çalışabilmesi için bir seçenek olarak programın içine yerleştirilir.

3 - PROGRAMIN ÇALIŞMA ŞEKLİ

Bu projede sistemdeki DOS Idle, seri kanal ve tuş kesmelerini kendi koduma yönlendirerek onlar üzerinde değişiklikler yaptım.

DOS Idle kesmesi DOS iş yapmıyorken belli aralıklarla çalışır. Bu kesme bilgisayarın çalışmasını kesmeden arka planda çalışmasını istediğiniz işlerin devamını sağlamak için çok uygundur. Tek sakıncası bazı programların çalışırken bu kesmeyi durdurmalarıdır.

Seri kanal kesmelerini bilgisayara ilgili seri kanaldan her karakter geldiğinde aktif olacak şekilde düzenledim. Böylece seri kanalı devamlı izlemeye gerek kalmadan, bu kanaldan gelen karakterler bilgi kaybına meydan vermeyecek şekilde alınabilir.

Tuş kesmesi her tuşa basıldığında aktif olur. Bu kesmeyi her tuşa basıldığında ilgili karakteri seri porttan atabilmek için kullandım.

DOS Idle kesmesini uzak noktadaki bilgisayarın ekran bilgisini aktarmak için kullandım. Program ilk çalıştığı anda tüm ekran bilgisini yakın bilgisayara atar. Ardından gelen her kesmeyle ekran farkları gönderilir. Böylece çok hızlı bir ekran tazelemesi mümkün olabilir. Gönderme işlemi seri port'a doğrudan karakter yazarak gerçekleşir, BIOS kullanılmaz..

Seri kanal kesmesi yakın bilgisayara gelen ekran bilgisi karakterlerinin değerlendirilip ekran görüntüsünün oluşturulması için kullanılır. Gelen bilgiden yerleşim yeri çözülür ve karakter uygun yere yerleştirilir.

Seri kanal kesmesi uzak bilgisayarda yakın bilgisayardan gelen tuş bilgilerinin işleme konulması için kullanılır. Gelen her tuş bilgisi uzak bilgisayardan basılmış gibi işleme konur. Böylece yakın noktadaki kullanıcı kendisini uzak noktadaki bilgisayarı kullanıyormuş gibi görebilir.

4 - PROJEDE İZLENEN YOL

Projeye başlamadan önce projeyi beş parçaya ayırdım.

4.1-Yakın PC'den uzak PC'ye yakın PC'de basılan tuş bilgilerinin

gönderilmesi

4.2-Uzak PC'den yakın PC'ye ekran değişimlerinin gönderilmesi

4.3-Uzak PC'nin kendisine gelen tuş bilgilerini algılaması

4.4-Yakın PC'nin kendisine gelen ekran bilgilerini algılaması

4.5-İletişimin modem üzerinden kurulması

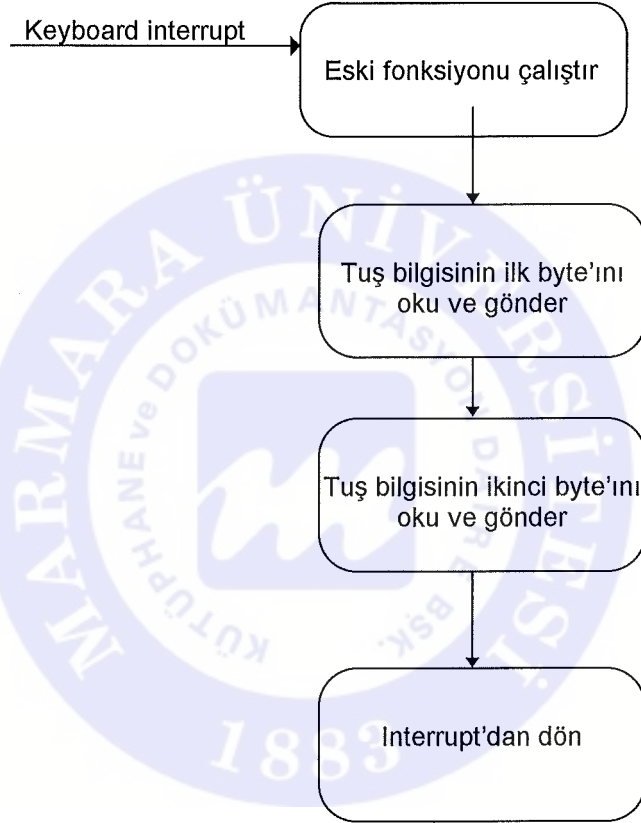
4.6-Proje parçalarının birleştirilmesi



4.1-Yakın PC'den uzak PC'ye yakın PC'de basılan tuş bilgilerinin gönderilmesi

Bir kişisel bilgisayarda her tuşa basıldığında tuş takımından ana kart'a o tuşa özel bir kod gelir(Scan Code). Bu kod BIOS'ta değerlendirilerek scan code'a karşılık düşen ASCII kod bulunur. Bu ASCII kod scan code'a göre karar verilen bir byte'lık bir ön kodla birlikte tuş takımının buffer'ına yazılır. İlgili bilgisayar üzerinde çalışan program ise bu buffer'ı okuyarak basılan tuşa bağlı ilgili işlemi yapar.

Yakın PC için geliştirdiğim program bu tuş bilgilerini buffer'dan okuyarak doğrudan seri port'a yazmaktadır.



Akış diyagramı yukarıda olan programın kodu Ek 1'de verilmiştir.

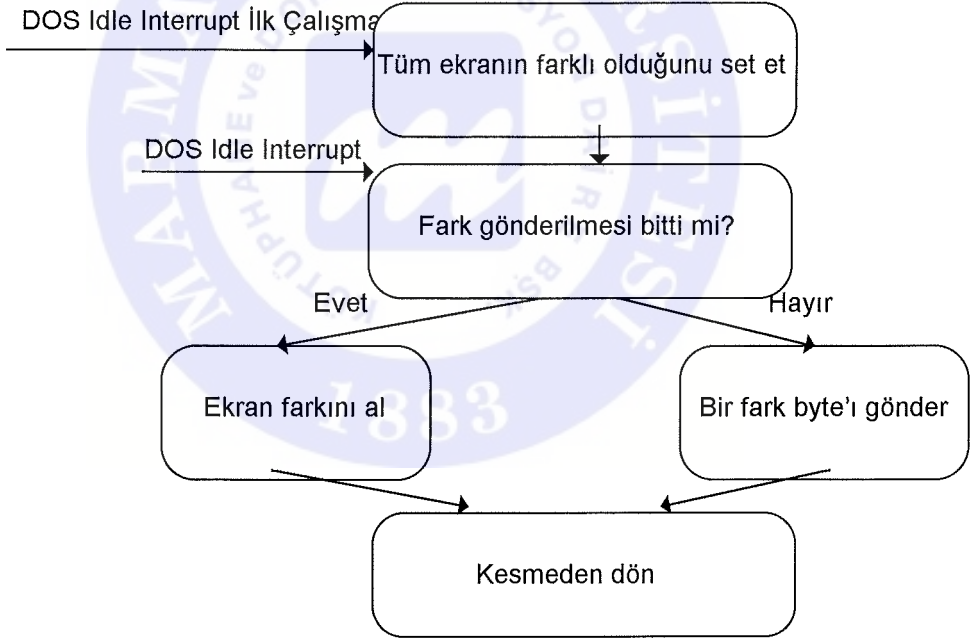
4.2 - Uzak PC'den yakın PC'ye ekran deęişimlerinin gönderilmesi

VGA kartı kullanan bir kişisel bilgisayar üzerinde 8 görüntü sayfası vardır. Bu sayfaların genelde ilk dördü kullanılmakta son dördü ise kullanılmamaktadır.

Uzak PC için geliştirdiğim program o andaki görüntü sayfasının bir kopyasını almakta ve orjinalle arasında bir fark oluştu mu diye izlemeye başlamaktadır. Fark oluştuğunu gördüğü anda farkları bir sayfaya depolamakta ve bu farkları tek tek seri kanaldan göndermektedir. Her fark bilgisi bir ön tanıma dizisi(FEH, FFH), yerleşim kodu(WORD) ve karakter kodundan oluşmaktadır. Program farkların gönderilmesi bittiği anda yeni fark oluştu mu diye tekrar ekran belleğini izlemeye başlamaktadır.

FE HEX	FF HEX	EKRAN POZİSYONU (WORD)	EKRAN BİLGİSİ (BYTE)
--------	--------	------------------------	----------------------

SERİ KANALDAN GİDEN EKRAN BİLGİSİ FORMATI

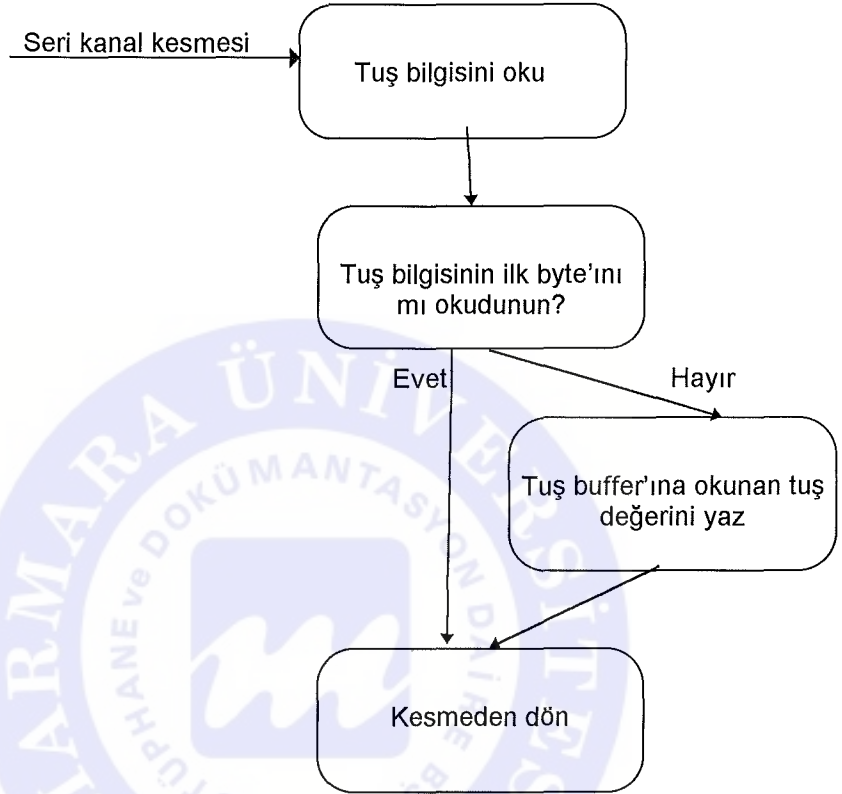


Akış Diyagramı yukarıda verilen programın kodu EK 3'dedir.

4.3 - Uzak PC'nin kendisine gelen tuş bilgilerini algılaması

Uzak noktadaki PC seri kanaldan gelen tuş bilgilerini değerlendirerek onları kendi tuş buffer'ına yerleştirir. Gelen her tuş bilgisi iki byte'tan oluşmaktadır.

Akış diyagramı aşağıda verilen programın kodu Ek 4'tedir

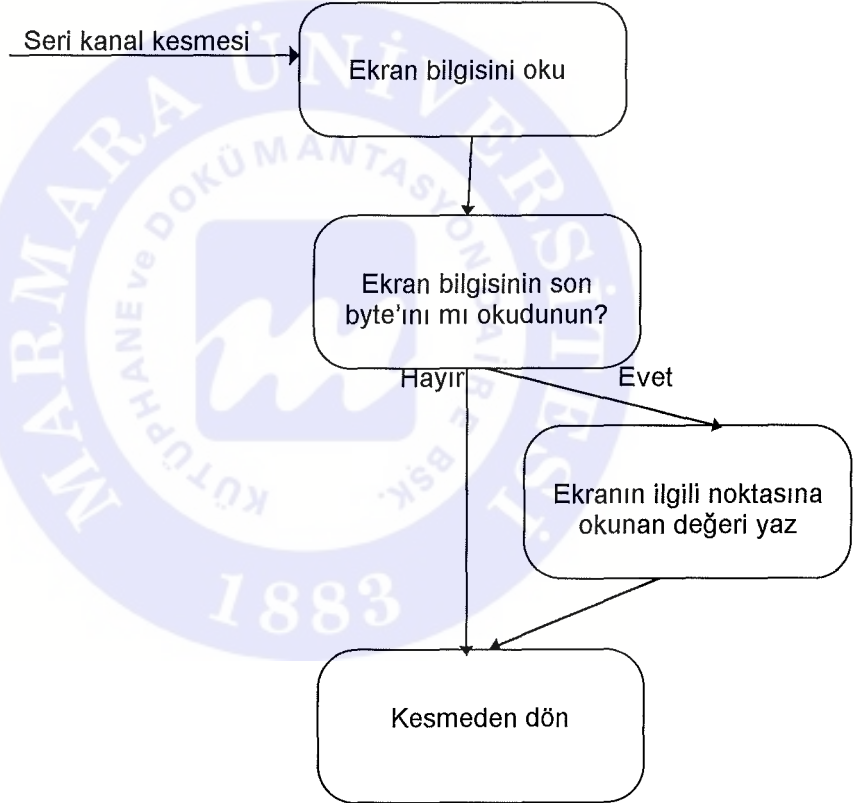


4.4 - Yakın PC'nin kendisine gelen ekran bilgilerini algılaması

Yakın noktadaki PC seri kanaldan gelen ekran karakter bilgilerini ekranın uygun noktalarına yerleştirir . Her karakter bilgisi bir ön tanıma dizisi(FEH, FFH), yerleşim kodu(WORD) ve karakter kodundan oluşmaktadır. Aşağıda akış diyagramı verilen programın kodu Ek 2'dedir.

FE HEX	FF HEX	EKRAN POZİSYONU (WORD)	EKRAN BİLGİSİ (BYTE)
--------	--------	------------------------	----------------------

SERİ KANALDAN GELEN EKRAM BİLGİSİ FORMATI



4.5 - İletişimin modem üzerinden kurulması

İki bilgisayarın birbirine yakın yerlerde bulunmaları durumunda aralarındaki bağlantı basit bir RS-232 kabloyla yapılabilir. Aradaki mesafe uzadıkça RS-232 protokolü hatasız bilgi iletişimi için yeterli ve ucuz bir yöntem olmamaktadır.

Modem adlı alet bilgisayarların haberleşmesi için dünya üzerindeki en eski ve en yaygın ağ olan telefon şebekesini kullanmaktadır. Bilgisayardan çıkan sayısal işaret modüle edilerek telefon hattına verilmekte hattın diğer ucunda bağlı bulunan modem ise bu işareti çözerek kendisine bağlı olan bilgisayara sayısal işaret olarak göndermektedir.

İlk modemler haberleşmeye 300 baud hızda başlamışlar, günümüzde standart olarak 28800 baud hızına kadar yükselmişlerdir. Günümüzde kullanılan modemlerin içine hata tanıma ve hata düzeltme kodları da yerleştirilmiştir.

İlk modemlerden itibaren Hayes modemi temel alan bir modem-bilgisayar haberleşme dili standart hale gelmiştir.

Bu projede aşağıdaki Hayes modem komutlarını kullanma gereği duydum.

ATZ0 : Sıfırlama. Kendi kendini test eder ve ayar profili 0'ı aktif kılar.

ATE0 : Komut konumunda karakter ekosuna izin verir.

ATQ1 : Modem sonuç kodlarını göndermez.

AT&D0 : Terminalden DTR işaretinin gelmesini beklemez.

ATD# : Numarayı çevirir.

ATS0=1: Telefon bir kere çaldıktan sonra hattı açar.

Modemlerden birisi aramak üzere diğeri ise arandığında hattı açmak üzere set edilir. Bağlantı kurulduktan sonra iki modem haberleşme hızlarına karar verirler ve modemin CD(Taşıyıcı algılandı) ledi yanar. Bu aşamadan sonra iki bilgisayarın arasına bir RS-232 kablo çekilmiş gibidir. Ve bir taraftan yollanan seri bilgi diğer taraftan alınmaya başlar.

Bu aşamadan sonra her iki noktada çalışması gereken programlar çalıştırılır. Ve ilk ekran aktarımı gerçekleşir.

4.6 - Proje parçalarının birleştirilmesi

Yukarıda anlatılan parçalar yapıldıktan sonra ilk modem bağlantısını düzenleyen programın ekran görüntüsü çekici hale getirildi ve bu program dışında yöneten ve yönetilen olmak üzere iki ayrı program oluşturuldu.

İlk olarak modem programı çalışmakta bağlantı kurulduktan sonra yöneten ve yönetilen kısımlardaki ilgili programlar çalışmaktadır.

4.7 - İleri çalışma önerileri

Günümüzde bu tip bir bağlantı DOS altında değil Windows altında yapılmaktadır. Bu bağlantıyı kurabilmek için en can alıcı nokta görüntünün karşı tarafa aktarılmasıdır. Bunu başarabilmek için Windows'un GDI katmanına ulaşmak gerekmektedir.

Bu tip bir işlem yine benim yaptığım programa benzer bir şekilde gerçekleştirilir. Yakın bilgisayarda tuş takımından gelen bilgiler yakalanıp seri porttan atılır. Bunlar için WINDOWS rutinleri mevcuttur. Seri porttan gelen bilgiler ise ekrana yazdırılır. Bu da WINDOWS kernel'da mevcuttur.

Yapılacak olan tek zor iş server tarafında WINDOWS GDI'ını capture edip buraya gelen tüm komutları seri porttan atmaktır.

Bunun dışında haberleşmenin güvenilirliğini sağlamak için bazı çerçeve bilgileri seri porttan giden bilgiye eklenebilir.

Ticari anlamda bir uygulama içinse user interface'ini güzelleştirmek gerekir.

5 - SONUÇ

Son kullanıcıya fazla iş bırakmadan kullanıcının kolay bir şekilde bir başka bilgisayara bağlanması ve buradaki bir programı çalıştırması sağlanmıştır. Bütün bu işlemler için kullanıcının sadece tek bir dosya çalıştırması yeterlidir.

Ortaya çıkan yazılım konuya ilgi duyan kişilerin üzerinde oynayabileceği zevkli bir koddur. Kişiler burada sunduğum mantığı kullanarak yakın noktadaki bilgisayarda gerçekte hiçbir şey yapmadan felaket sahnelerinin ekrana gelmesini sağlayabilirler. Veya tam tersi olarak uzak noktadaki bilgisayarda bir anda bir yazılımın çalışmaya başlamasını sağlayabilirler.

İnsan olmayan noktalardaki bilgisayarların yönetimi için de bu tip yazılımlar kullanılmaktadır. Örneğin kutuplarda bir meteoroloji noktasındaki bilgisayar bu tip bir yazılımla yönetilebilir.

6 - Kaynaklar

- * The New Peter Norton's PROGRAMMER'S GUIDE TO The IBM PC & PS/2
By Peter Norton & Richard Wilton, 1988 Microsoft Press
- * Data Communications, Computer Networks And OSI
By Fred Halsall, 1988 Addison-Wesley Publishing Company
- * Byte International ve Byte Türkiye dergilerinin çeşitli sayıları

7 - EKLER

EK 1 Yöneten bilgisayarda basılan tuşları gönderen

program

```
;* Keybrd - Interrupt handler for Interrupt 09 (keyboard).
;*
;* IBM PC/AT and compatibles:
;*   Gets the scan code of the current keystroke from port 60h. Then
;*   compares the scan code and shift state to the hot key. If they
;*   match, sets TsrRequestFlag to signal the handlers Clock and Idle
;*   that the TSR is requested.
;*
;* IBM PS/2 series:
;*   Only the instructions at KeybrdMonitor (see below) are installed
;*   as Interrupt 09 handler, since above method should not be used to
;*   determine current keystroke in IBM PS/2 series. In this case, the
;*   Interrupt 15h handler MiscServ takes care of checking the scan
codes
;*   and setting the request flag when the hot key is pressed.
;*
;* Time-activated TSRs:
;*   If the TSR is activated by time instead of by a hotkey, KeybrdMonitor
;*   serves as the Interrupt 09 handler for both PC/AT and PS/2 systems.
;*
;* Uses:  intKeybrd, TsrRequestFlag
;*
;* Params: None
;*
;* Return: None
```

Keybrd PROC FAR

```
    sti                ; Interrupts are okay
    push ax           ; Save AX register
    in  al, 60h       ; AL = scan code of current key

    call CheckHotKey ; Check for hot key
    .IF !carry?      ; If not hot key:

        pushf
        pusha
        cli
        INVOKE Deinstall ; Unchain interrupt handlers
        .IF ax > OK_ARGUMENT ; If successful,
            INVOKE FreeTsr, ; deinstall TSR by freeing
memory      ax ; Address of resident seg
        .ENDIF ; Else exit with message
        popa
        popf
        mov  al, 20h ; Reset interrupt controller
        out 20h, al
        sti
        pop  ax ; Recover AX
        iret ; Exit interrupt handler
    .ENDIF ; End hot-key check

; No hot key was pressed, so let normal Int 09 service routine take over
    pop  ax ; Recover AX and fall through
    cli ; Interrupts cleared for service
```

```

KeybrdMonitor LABEL FAR          ; Installed as Int 09 handler for
                                   ; PS/2 or for time-activated TSR

mov  cs:intKeybrd.Flag, TRUE ; Signal that interrupt is busy
pushf                                ; Simulate interrupt by pushing flags,
call cs:intKeybrd.OldHand  ; far calling old Int 09 routine

```

```

pusha

```

```

don:

```

```

mov  ah, 11h
int  16h
jz   devamkeyread
mov  ah, 10h
int  16h
push ax

```

```

gonderme:

```

```

mov  dx, cs:comportadr  ;Check if the serial port is empty
add  dx, 5
in   al, dx
and  al, 00100000y
cmp  al, 00100000y
jz   gonder
jmp  gonderme

```

```

gonder:

```

```

pop  ax
push ax

```

```

        mov     dx, cs:comportadr    ; Send the first byte of keyboard
info
        out     dx, al

gonderme2:
        mov     dx, cs:comportadr    ; Check if the serial port is empty
        add     dx, 5
        in     al, dx
        and     al, 00100000y
        cmp     al, 00100000y
        jz     gonder2
        jmp    gonderme2

gonder2:
        pop     ax
        mov     al, ah
        mov     dx, cs:comportadr    ; Send the first byte of keyboard
info
        out     dx, al

devamkeyread:
        popa
        mov     cs:intKeybrd.Flag, FALSE
        iret

Keybrd ENDP

```

Ek 2 Yöneten bilgisayara gelen ekran bilgilerini algılayan program

PortIO PROC FAR

```
sti
pushf
push  bx
push  ax
push  cx
push  dx
push  ds
push  es
push  si
push  di
push  cs
pop   ds           ; Set DS to resident code segment
ASSUME ds:@code
mov   intCom.Flag, TRUE   ; Set active flag
```

; Read Char

```
mov   dx,ComPortAdr     ;load Comm.port Adress
in    al,dx             ;read data
pusha
cmp   al, 0FEh         ;0FEh, 0FFh olan baslangic
```

sensorlerini

```
jz    lookatbegin1     ;ara
cmp   al, 0FFh
jz    lookatbegin2
```

gitswtch:

```

cmp      cs:chrswitch, 1      ; Get x coordinate
jz       clyukle
cmp      cs:chrswitch, 2      ; Get y coordinate
jz       chyukle
cmp      cs:chrswitch, 3      ; Get char
jz       charyukle
jmp      vidyukleson

```

lookatbegin1:

```

mov      cs:beginsense, 1
cmp      chrswitch, 1
jz       gitswitch
cmp      chrswitch, 2
jz       gitswitch
cmp      chrswitch, 3
jz       gitswitch
jmp      vidyukleson

```

lookatbegin2:

```

cmp      cs:beginsense, 1
jnz      gitswitch
mov      cs:chrswitch, 1
mov      cs:beginsense, 2

```

ciksense:

```

jmp      vidyukleson

```

clyukle:

```

mov      cs:vidcl, al
mov      cs:chrswitch, 2
jmp      vidyukleson

```

chyukle:

```

mov cs:vidch, al
mov cs:chrswitch, 3
jmp vidyukleson

```

charyukle:

```

mov cs:vgachar, al
mov cs:chrswitch, 0
mov ax, 0b800h
mov es, ax
mov bl, cs:vidcl
mov bh, cs:vidch
mov al, cs:vgachar
cmp bx, 0e00eh ;Set Cursor row position

```

command ?

```

jz rowpospl
cmp bx, 0efeeh ;Set Cursor column position

```

command ?

```

jz colpos
mov es:[bx], al ;Char yaz
jmp vidyukleson

```

rowpospl:

```

mov cs:rowpos, al ;Set Cursor row

```

position command

```

jmp vidyukleson

```

colpos:

```

pusha
mov cl, al
mov bx, 0062h ;Set Cursor column position

```

command

```

mov ax, 0040h
mov es, ax

```

```

mov     al, es:[bx]
mov     bh, al
mov     dh, cs:rowpos
mov     dl, cl
mov     ah, 02
int     10h

popa

jmp     vidyukleson

vidyukleson:
        popa
; Send Ond Of Interrupt

notLegal:
        mov     al,20h           ;send EOI to 8259
        out    PIC_EOI,al
        mov     intCom.Flag, FALSE ;reset active flag
        pop     di
        pop     si
        pop     es
        pop     ds
        pop     dx
        pop     cx
        pop     ax
        pop     bx

        popf

        iret

PortIO ENDP

```

Ek 3 Yönetilen bilgisayarın ekran değişimlerini gönderen

program

```
;* Idle - Interrupt handler for Interrupt 28h (DOS Idle). Allows the  
;* original Interrupt 28h service routine to execute. Then checks the  
;* request flag TsrRequestFlag maintained either by the keyboard handler  
;* (keyboard-activated TSR) or by the timer handler (time-activated TSR).  
;* See header comments above for Clock, Keybrd, and MiscServ  
procedures.
```

```
;*
```

```
;* If TsrRequestFlag = TRUE and system is in interruptable state, Idle  
;* invokes the TSR by calling the Activate procedure. Uses an active flag  
;* to prevent the Idle procedure from being reentered while executing.
```

```
;*
```

```
;* Uses: intIdle and TsrActiveFlag
```

```
;*
```

```
;* Params: None
```

```
;*
```

```
;* Return: None
```

```
Idle PROC FAR
```

```
    pushf                ; Simulate interrupt by pushing flags,
```

```
    call cs:intIdle.OldHand ; far-calling old Int 28h routine
```

```
    .IF cs:intIdle.Flag == FALSE; If not already in this handler,
```

```
    mov  cs:intIdle.Flag, TRUE ; set active flag
```

```
    sti                ; Interrupts are okay
```

```
    push ds            ; Save application's DS
```

```

push cs
pop ds ; Set DS to resident code segment
ASSUME ds:@code
pushf
pusha

;;; *** mov page 1 to page 8
;current page`i bul
pusha
push ds
mov ah, 0Fh
int 10h ; Get current mode
mov ax, 0B800h ; video buffer
add ah, bh ; Adding display page
gives
pop ds
mov ds:vidsegmt, ax ; address of current page
popa

;*****COMPARE PAGE 1 WITH PAGE 4
cmp cs:sendok, 0
jnz chargond
cmp cs:ilksetup, 0
jnz ilkdegil

; ilksetup
mov cs:ilksetup, 1
cmp cs:ilkekrangonder, 0
jz ilkekrangnd
jmp cikcmp

ilkdegil:

```

;;sendingnotok ise sayfa karsilastir

```
mov     cs:farkvar,0
mov     cx, 0
mov     si, 0
mov     di, 07000h
mov     ax, 0b800h
```

devamcmp1:

```
push    ds
mov     ax, 0b800h
mov     ds, ax
mov     bx, di
add     bx, cx
mov     al, ds:[bx] ;b800:07000h
pop     ds
mov     cs:chsakla, al
mov     ax, cs:vidsegmt
push    ds
mov     ds, ax
mov     bx, si
add     bx, cx
mov     bl, ds:[bx] ;b900:00000h
pop     ds
mov     al, cs:chsakla
cmp     bl, al
jnz     farkli
mov     ax, 0b800h
push    ds
mov     ds, ax
mov     bx, cx
add     bx, 06000h ; page 7
```

```
mov     dl, 0
mov     ds:[bx], dl
pop     ds
```

devamartir:

```
inc     cx
cmp     cx, 3999
jnz     devamcmp1
mov     al, cs:farkvar
mov     cs:sendok, al
mov     cs:vgasendcnt, 0
```

;***Buradayim mesaji

```
pusha
```

;current page'i 8. sayfaya at

```
mov     cx, 3999
mov     si, 0
mov     di, 07000h
mov     ax, 0b800h
mov     es, ax
mov     ax, cs:vidsegmnt
mov     ds, ax
rep     movsb
popa
jmp     cikcmp
```

farkli:

```
push   ds
mov     ax, 0b800h
mov     ds, ax
mov     bx, cx
add     bx, 06000h           ; page 7
mov     dl, 1
```

```
mov     ds:[bx], dl
mov     cs:farkvar, 1
pop     ds
jmp     devamartir
```

;***** ilkekrangonder

ilkekrangnd:

```
pusha
```

;current page'i 8. sayfaya at

```
mov     cx, 3999
mov     si, 0
mov     di, 07000h
mov     ax, 0b800h
mov     es, ax
mov     ax, cs:vidsegmt
mov     ds, ax
rep     movsb
popa
mov     cx, 0
```

devamilkekr:

```
push    ds
mov     ax, 0b800h
mov     ds, ax
mov     bx, cx
add     bx, 06000h
mov     dl, 1
mov     ds:[bx], dl
pop     ds
inc     cx
cmp     cx, 3999
jnz     devamilkekr
```

; page 7

```

mov     cs:ilkekrangonder, 1
mov     cs:sendok, 1
mov     cs:vgasendcnt, 0
jmp     cikomp
;***** char gonder

```

chargond:

;buradayim mesaji

```

mov     dx, comportadr
add     dx, 5
in      al, dx
and     al, 00100000y
cmp     al, 00100000y
jnz     chargond

```

;buradayim mesaji

```

push   ds
mov     ax, 0b800h
mov     ds, ax
mov     bx, 06000h
add     bx, cs:vgasendcnt
mov     al, ds:[bx]
pop     ds
cmp     al, 1
jz      gonderok
inc     cs:vgasendcnt
mov     ax, 3999
cmp     cs:vgasendcnt, ax
jnz     cikomp
mov     cs:sendok, 0
mov     cs:vgasendcnt, 0
mov     cs:ilksetup, 0

```

;cursor place gonder

crsrplcgond:

bosaltyer:

```
mov dx, comportadr
add dx, 5
in al, dx
and al, 00100000y
cmp al, 00100000y
jnz bosaltyer
mov al, 0FEh
mov dx, comportadr
out dx, al
```

bosaltyer0:

```
mov dx, comportadr
add dx, 5
in al, dx
and al, 00100000y
cmp al, 00100000y
jnz bosaltyer0
mov al, 0FFh
mov dx, comportadr
out dx, al
```

bosaltyer2:

```
mov dx, comportadr
add dx, 5
in al, dx
and al, 00100000y
cmp al, 00100000y
jnz bosaltyer2
mov al, 0eeh
```

```
mov dx, comportadr
out dx, al
```

bosaltyer3:

```
mov dx, comportadr
add dx, 5
in al, dx
and al, 00100000y
cmp al, 00100000y
jnz bosaltyer3
mov al, 0eeh
mov dx, comportadr
out dx, al
```

bosaltyer4:

```
mov dx, comportadr
add dx, 5
in al, dx
and al, 00100000y
cmp al, 00100000y
jnz bosaltyer4
mov bx, 0062h
mov ax, 0040h

push ds

mov ds, ax
mov al, ds:[bx]
add al, al
mov bx, 0050h
mov ah, 0
add bx, ax
mov ax, ds:[bx]
mov al, ah
```

```
mov dx, comportadr
out dx, al
pop ds
```

bosaltyer02:

```
mov dx, comportadr
add dx, 5
in al, dx
and al, 00100000y
cmp al, 00100000y
jnz bosaltyer02
mov al, 0FEh
mov dx, comportadr
out dx, al
```

bosaltyer20:

```
mov dx, comportadr
add dx, 5
in al, dx
and al, 00100000y
cmp al, 00100000y
jnz bosaltyer20
mov al, 0FFh
mov dx, comportadr
out dx, al
```

bosaltyer22:

```
mov dx, comportadr
add dx, 5
in al, dx
and al, 00100000y
cmp al, 00100000y
jnz bosaltyer22
```

```

mov     al, 0eeh
mov     dx, comportadr
out     dx, al

bosaltyer23:
mov     dx, comportadr
add     dx, 5
in      al, dx
and     al, 00100000y
cmp     al, 00100000y
jnz     bosaltyer23
mov     al, 0efh
mov     dx, comportadr
out     dx, al

bosaltyer24:
mov     dx, comportadr
add     dx, 5
in      al, dx
and     al, 00100000y
cmp     al, 00100000y
jnz     bosaltyer24
push   ds
mov     bx, 0062h
mov     ax, 0040h
mov     ds, ax
mov     al, ds:[bx]
add     al, al
mov     bx, 0050h
mov     ah, 0
add     bx, ax
mov     ax, ds:[bx]

```

```
mov dx, comportadr
out dx, al
pop ds
jmp cikcmp
```

gonderok:

```
;bosalt
```

bosalt:

```
mov dx, comportadr
add dx, 5
in al, dx
and al, 00100000y
cmp al, 00100000y
jnz bosalt
mov al, 0FEh
mov dx, comportadr
out dx, al
```

bosalt0:

```
mov dx, comportadr
add dx, 5
in al, dx
and al, 00100000y
cmp al, 00100000y
jnz bosalt0
mov al, 0FFh
mov dx, comportadr
out dx, al
```

bosalt2:

```
mov dx, comportadr
add dx, 5
in al, dx
```

```
and        al, 00100000y
cmp        al, 00100000y
jnz        bosalt2
mov        ax, cs:vgasendcnt
mov        dx, comportadr
out        dx, al
```

bosalt3:

```
mov        dx, comportadr
add        dx, 5
in         al, dx
and        al, 00100000y
cmp        al, 00100000y
jnz        bosalt3
mov        ax, cs:vgasendcnt
mov        al, ah
mov        dx, comportadr
out        dx, al
```

bosalt4:

```
mov        dx, comportadr
add        dx, 5
in         al, dx
and        al, 00100000y
cmp        al, 00100000y
jnz        bosalt4
push      ds
mov        bx, 07000h
add        bx, cs:vgasendcnt
mov        ax, 0b800h
mov        ds, ax
mov        al, ds:[bx]
```

```

mov     dx, comportadr
out     dx, al
pop     ds
inc     cs:vgasendcnt
mov     ax, 3999
cmp     cs:vgasendcnt, ax
jnz     cikcmp
mov     cs:sendok, 0
mov     cs:vgasendcnt, 0
mov     cs:ilksetup, 0
jmp     crsrplcgond
cikcmp:
popa
popf
cikkk:
mov     cs:intIdle.Flag, FALSE    ; Clear active flag
pop     ds                        ; Recover application's DS
.ENDIF                             ; End in-handler check

iret
Idle   ENDP

```

Ek 4 Yönetilen bilgisayara gelen tuş bilgilerini algılayan

program:

PortIO PROC FAR

sti

pushf

push bx

push ax

push cx

push dx

push ds

push es

push si

push di

push cs

pop ds ; Set DS to resident code segment

ASSUME ds:@code

mov intCom.Flag, TRUE ; Set active flag

; Read Char

mov dx,ComPortAdr ;load Comm.port Adress

in al,dx ;read data

cmp al, 0

jnz noresetkey

mov chrswitch, 0

noresetkey:

cmp chrswitch, 0

jz asciiCd

mov cl, asciiCode

mov ch, al

mov ah, 05h

```

        int     16h
        mov     chrswitch, 0
        jmp     asciiscanson

asciicd:
        mov     asciicode, al
        mov     chrswitch, 1

asciiscanson:
        jmp     notLegal

; Send End Of Interrupt

notLegal:
        mov     al,20h           ;send EOI to 8259
        out     PIC_EOI,al
        mov     intCom.Flag, FALSE ;reset active flag
        pop     di
        pop     si
        pop     es
        pop     ds
        pop     dx
        pop     cx
        pop     ax
        pop     bx
        popf
        iret

PortIO ENDP

```