mikrobilgisayarların dış dünya ile iletişimi -giriş/çıkış birimleri-

Kaya YAZGAN

UDK: 621.3.049.771.14: 621.3.049.61:621.3.049.62

ÖZET

Giriş/Çıkış birimleri mikroişlemcili sistemlerin en önemli yapıtaşlartndan birini oluşturmaktadır. Bu kısaltılmış çeviri çalışmasında, Programlı. Kesmeli ve Dolaysız Bellek Erişimli giriş/çıkış işlemleri seri giriş/çıkış kavram ve birim düzeylerinde tanıtılmaktadır. Temel olarak sekiz bitlik bir mikroişlemci dizgesi alınmış ve elden geldiğince genel amaçlı giriş/çıkış birimleri oluşturulmuş ve tartışılmıştır.

SUMMARY

Input/Output Unit is one of the most important building blocks of the microprocessor systems. in thit short translation, programmed, interrupted and direct memory acces», serial input/output are introduced as concepts and modüle*. As a basis an 8-bit microprocessor system v conâdered and general purpose input/output modules are designed and discussed.

Mikroişlemciyle birlikte çalışmak için tasarımlanmış mantık devrelerinin tümüne mikrobilgisayar dersek, bu dizgenin dışında kalan herşeyle iletişim sağlamak görevini üstlenen arabirimlere de giriş/çıkış birimleri adını verebiliriz. Genel olarak veri iletişiminin sağlanması için birçok yöntem düşünülebilirse de tümünü üç ana grup içinde değerlendirme olanağı vardır.

Kaya Yazgan, ASELSAN

t) Programlı Giriş/Çıkış uygulamalarında giriş/çıkış işlemleri mikroişlemcide izlenen programın gereklerine uygun olarak yapılır.

2) Kesmeli *(Interrupt)* Giriş/Çıkış uygulamalarında ise Um tersine mikrobilgisayar dışı mantık birimleri mikroişlemcinin yürüttüğü programı kesip bir süre bekletirler.

3) Dolaysız Bellek Erişimi *(Direct Memory Access, DMA)* özelliğine sahip giriş/çıkış işlemlerinde dış iletişim aygıtları ile mikrobilgisayar mantığı arasındaki veri alış-verişi merkez işlem biriminden geçmeden, doğrudan doğruya gerçekleşir.

Bu incelemede önce herbir iletişim türü tartışılacak, ardından da seri Giriş/Çıkış konusunda bazı bilgiler sunulacaktır.

PROGRAMLI GİRİŞ/ÇIKIŞ

Verinin mikrobilgisayar dizgesi dışına çıkışı giriş/çıkış kapıları *(port)* adı verilen devreler aracılığıyla yapılır. Sekiz bitlik bir mikroişlemci dizgesinin 8 iletkeni! bir veri yolu *(data bus)* ve 16 iletkenli bir adres *yo\u(address bus)* olduğu varsayılırsa 40 bacaklı koşut *(parallel)* bir giriş/çıkış biriminin

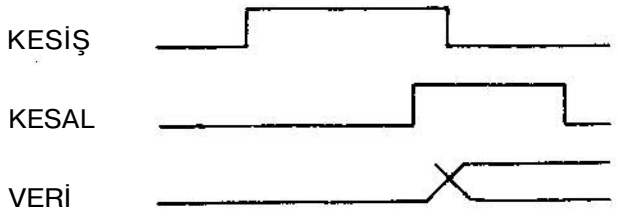
- **16 bacağı adres yolu**
- **3 bacağı kaynaklar ve toprak**
- 8 bacağı veri yolu (mikrobilgisayar yönü)
- 8 bacağı dış devrelere veri gönderen uçlar
- 1 bacağı saat
- 1 bacağı "yaz" komutu
- 1 bacağı "oku" komutu

için ayrılırsa geriye yalnızca 2 boş bacak kalmaktadır. Böyle bir dizgede örneğin $(7\ FFF)_{16}$ ve daha küçük adresler bellek adresi, $(8000)_{16}$ ve üstü ise giriş/çıkış kapılarının adreslerine ayrılabilir Bu durumda $2^{15} = 32768$ tane giriş/çıkış kapısını adresleme olanağına kavuşuruz ki hemen hemen hiçbir uygulamada bu denli çok sayıda giriş/çıkış birimine gerek yoktur. Buna karşılık iki kapının bir tümleşik devreye sığdırılması çok büyük yararlar sağlayacaktır. O zaman iki kapılı giriş/çıkış birimlerinde

- 10 bacak adres yolu
- 3 bacak kaynaklar ve toprak
- 8 bacak veri yolu (mikrobilgisayar yönü)
- 8 bacak A kapısı (bir çevre aygıtına)
- 8 bacak B kapısı (başka bir çevre aygıtına)
- 1 bacak saat
- 1 bacak "yaz" komutu
- 1 bacak "oku" korr.uu ı

olarak belirlenebilir. Yine 15 inci adres iletkeni " 1 " iken bir giriş/çıkış biriminin seçildiğini, bunun dışında (A0-AS) den oluşan 9 iletkenin daha giriş/çıkış birimlerine bağlandığını düşünelim. (8000)i6 ile (81FF)i6 arasındaki 256 adres giriş/çıkış birimlerine ayrılacaktır. Şimdiye kadar değindiğimiz iki örnekte de A15'in giriş/çıkış birimlerine ayrılması nedeniyle adreslenebilir (2^{16} – 65 536 = 65 K) sözcıklük bellekten vazgeçip (2^{15} = 32 768 = 32 K) sözcüklü bellek kullanılmaktadır. Pekçok mikroişlemci uygulamasının 1024 - 4096 sözcüklük bellek büyüklükleri içinde kalması bu azalmanın önemli bir sakınca olmadığını gösterir.

Bellek büyüklüğünün 32 K'da sınırlanamadığı durumlarda ise ayrı giriş/çıkış adres mantığı bulunan dizgelere gerek vardır, örneğin GÇSEÇ ve GÇOY gibi iki denetim bacağına ek olarak 7 adres iletkeni olan giriş/çıkış yon-

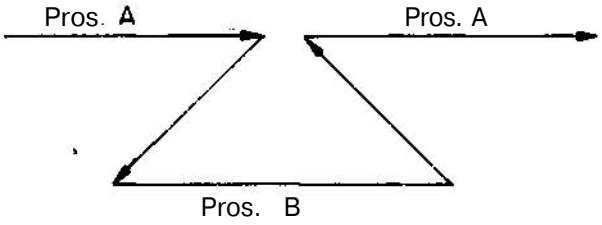


Şekil 1.

galan vardır. GÇSEÇ AO-A7 iletkenlerinin bir kapı adreslediğini verirken GÇOY'ın yüksek olması okumayı, alçak olması yazmayı belirleyebilir. Çoğu kez bu tür bir denetim de yetersizdir. Giriş/çıkış birimi, hem çevre aygıtı ile hem de mikrobilgisayarla denetim ilişkisi kurmak zorundadır. Mikrobilgisayardan dışarıya giden bilgiye giriş/çıkış denetimi *(I/O control);*dışardan mikrobilgisayara yönelen bilgiye ise giriş çıkış konumu *(I/O status)* adı verilir.

KESMELİ GİRİŞ/ÇIKIŞ

Birçok mikroişlemci uygulamalarında çevre aygıtlarından gelen giriş/çıkış istemlerinin elden geldiğince çabuk değerlendirilmesi gerekir. Programın belirli noktalarında bu çevre birimlerinin istemi olup olmadığı incelenebilirse de geç kalınması ya da, gereksiz yere sık sık incelenmesi nedeniyle programın çok yavaşlaması tehlikeleri vardır. Bu durumda Şekil 1 "de görüldüğü gibi giriş/çıkış biriminin mikroişlemciye KESİŞ, kesme istemi *(IREQ, interrupt request)* göndermesi, ardından mikroişlemcinin kısa sürede KESAL, kesme alındı *(IACK, interrupt acknoui-lcdge)* bilgisini vermesi ve veri *(DATA)* iletiminin başla-



Sekil *2.*

ması sağlanır. Mikroişiemcinin yürütmekte olduğu program açısından bakmca,programın kesilip veri iletiminin yapıldığı,sonra kalan yerden programa dönüldüğü gözlenir. (Şekil 2).

Mikroişlemci bir kesme istemi aldığında,denetim birimi *(control unit)* içinde hazır olan bir mikroprogram aracılığıyla, istemi aldığı noktada tüm yazmaçların *(register)* içindeki bilgileri belleğe yükler. Bu arada yürüttüğü programın hangi noktada olduğunu veren program sayacı *(program counterj* içindeki bilginin de saklanması sonradan aynı noktadan devam etme olanağı sağlar. Mikroprogram biçiminde bir yazılım dizgede yoksa programcı da bu amaçla program yazabilir.

Kesme süresince yapılması gereken işlerin neler olduğunun da, kuşkusuz, mikroişlemciye bir program şeklinde verilmesi gereklidir. Bu da çoğunlukla giriş/çıkış birimlerindeki bi; yazmaç kesme adres vektörünün *(interrupt address ver tor)* gösterdiği bir bellek adresinden bulunur. Mikroişlemcinin KESAL imi göndermesinden sonra giriş/çıkış birimi adres yolu üzerinden adres vektörünü merkez işlem birimine gönderir. Giriş/çıkış birimine giren adres iletkenlerinin sayısının tam bir adres vermeye yetmemesi durumunda (daha önce 16 yerine 7-10 iletken kullanılabildiğine değinmiştik) vektör ikiye bölünüp iki ayrı saat vuruşuyla gönderilebilir. Çoğu kez kesme programı bir salt oku bellektedir ve adres vektörü de bu programın başlangıç adresini taşımaktadır.

Kesme önceliği

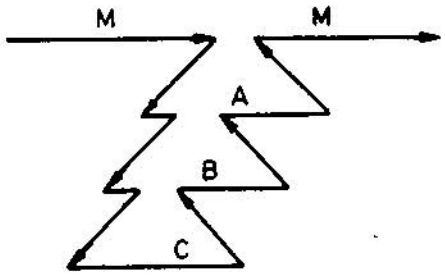
Birden fazla giriş/çıkış biriminin olduğu uygulamalarda birkaç birim aynı anda ya da birbirinin hemen ardından kesme isteminde bulunabilirler. Varsayalım ki M prog-

Şekil 3.

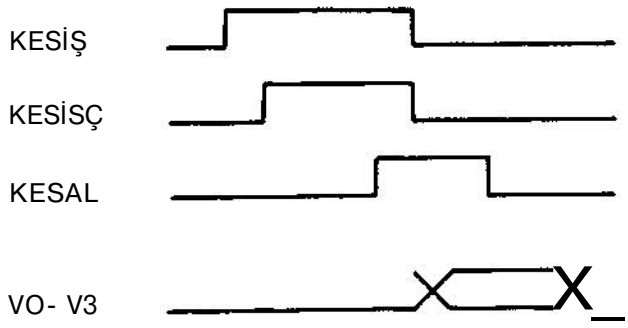
ramı yürürken aynı anda A, B ve C programlarının yürütülmesini isteyen üç kesme istemi gelsin. En büyük önceliğe sahip istemin A olduğunu, sonra da öncelik sırasıyla B ve C'nin geldiğini düşünelim. Programın Şekil 3'de görüldüğü gibi yürümesi amacımızdır. Eğer A'nm yürümesi başlayınca mikroişlemciye B'nin ve C'nin KESİŞ göndermesi engellenmezse A programı KESAL aldığı için KESİS'ini kaldıracak bu kez B'nin KESİŞİ olduğundan A programı bitmeden B'ye, o da bitmeden C'ye geçilecektir (Şekil 4). Kesme önceliği önemli bir sorun oluşturursa kesme öncelik birimi *(interrupt priority urdt)* kullanılabilir. Örnek olarak 16 giriş/çıkış birimini denetleyen bir öncelik birimi düşünelim. Herbir giriş/çıkış biriminin KESİŞ iletkenleri öncelik biriminin ayrı birer bacağına bağlanacaktır. Bağlanma sıraları önceliklerini belirlemede yeterlidir.Şekil 5'de görüldüğü gibi bir ya da birkaç giriş/çıkış biçiminin KESİŞ göndermesi üzerine öncelik devresi mikroişlemciye KESİSÇ (kesme istemi çıkış) gönderir. Mikroişlemciden gelen KESAL üzerine ise öncelik devresi VO - V3 iletkenlerinden kesme isteminde bulunan en öncelikli birimin hangisi olduğunu bildirir. Bundan sonra giriş/çıkış birimi ile mikroişlemci karşı karşıya kalıp daha önce özetlendiği gibi iletim yaparlar.

Zincirleme Yöntemi

Mikroişlemcili dizgede yalnızca bir kesme hattı bulunması durumunda zincirleme yöntemi *(daisy chain)* adı verilen yönteme başvurulur. Şekil 6'da görüldüğü gibi

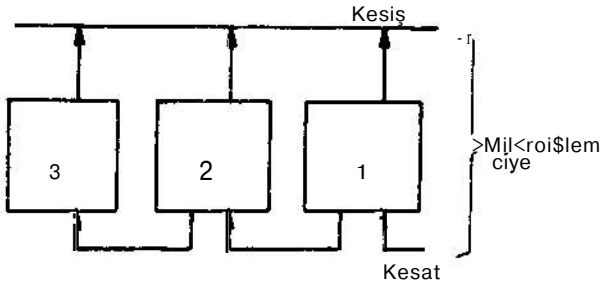


Şekil 4.



Şekil 5.

aygıtlardan herbiri tek KESİŞ iletkenine bağlıdır. Bir ya da birkaç aygıt kesme isteğinde bulunursa hangisinin istemde bulunduğu mikroişlemci tarafından bilinmez. Ama mikroişlemcinin gönderdiği KESAL'ın zincirleme hepsini dolaştığı ve kesme istemi olmayan aygıtın KESAL imini bir sonrakine geçirmesi, istem sahibinin ise kesme işlemine başlaması sağlanırsa öncelik sorunu pratik bir çözüme kavuşur.



Şekil 6.

Kaynak Kesilmesi

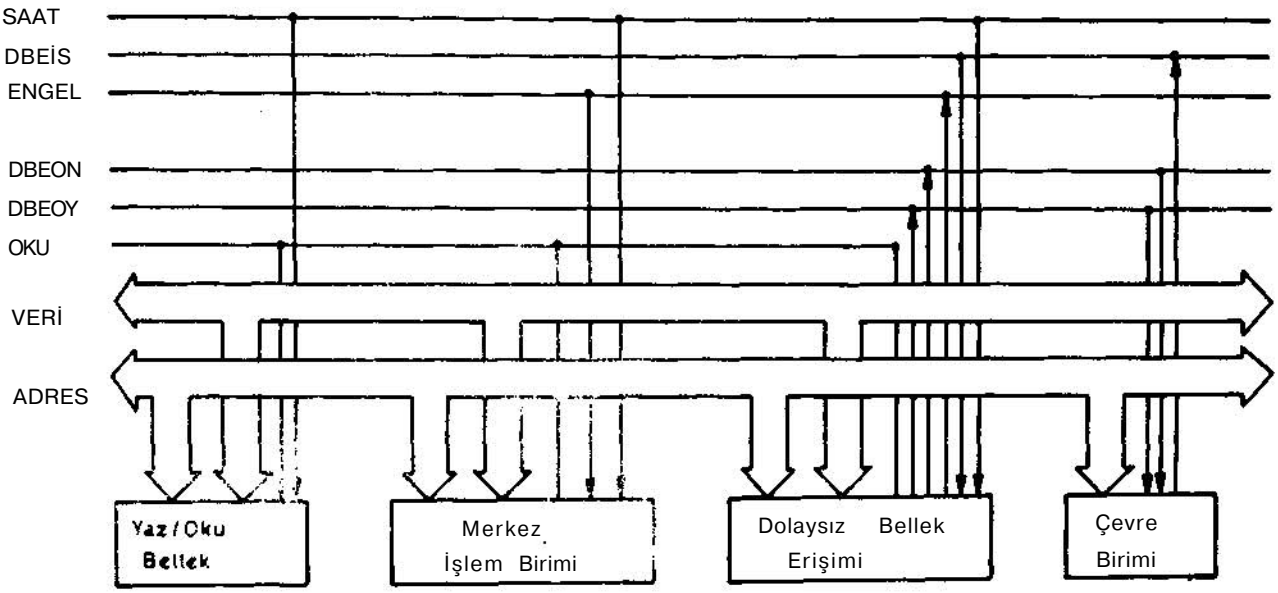
Bu tür bir zincire ilk bağlanacak - en öncelikli - kesme kaynak hakkındaki bilgidir. Birçok mikroişlemci + 5V ve/veya + 12 V DA kaynağından beslenmektedir. Bir güç kaynağı 110 ya da 220 V AA'dan bu DA gerilimleri elde etmektedir. Şebeke gerilimi anma değerinin % 20 altına düşünce bir kesme veren devre tasarlanıp dizgeye katılırsa + 5 ve + 12 V kaynaklar dizgeyi süremez hale gelinceye dek geçecek saniyenin binde biri düzeyindeki süre içinde yüzden fazla mikrobilgisayar komutunu işlemek olanağı vardır. Bir kesme programı bu süre içinde yazmaçları uçucu olmayan *(nonvolotile)* yani gerilim kesilince kapsadığı bilgiyi yitirmeyen bir belleğe boşaltmakta ve gerilim gelince de aynı noktadan devam etmektedir.

DOLAYSIZ BELLEK ERİŞİMİ

Bir kesme istemi geldiğinde ardarda yapılması gereken işlemleri en yalın biçimde özetlersek;

1) Mikroişlemci bir A programını yürütmekte iken kesme istemi gelirse tüm yazaçlar belleğe aktarılır
2) istemin gereği olan B programı yürütülür
3) Yeniden A programına kalınan yerden devam etmek için yazmaçlar saklanan sayılarla doldurulur. Program sayacı bir ilerletilir.

Dikkat edilirse (1) ve (3) adımlarının ne tür kesme istemi gelirse gelsin yerine getirildiği, pek karmaşık olmayan dizgelerde (2) adımınında hep yinelenen bir program olduğu gözlenir. En basit bir okuma işlemi için bile mikroişlemci 50 mikrosaniye dolayında zaman yitirir. İşte bu sakıncayı önlemek için çevre birimleri ile bellek arasında



Şekil 7.

mikroişlemciden geçmeyen bir veri yolu kurulur. Bu yolu denetlemek için de özel bir tümleşik devre dolaysız bellek erişim ***(Direct Memory Access)*** kullanılır.

Çevrim Çalma

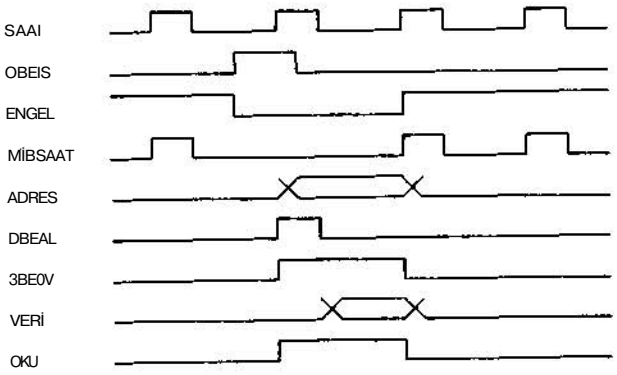
En çok kullanılan dolaysız bellek erişimi yöntemi merkez işlem birimini bir çevrim ***(eyele)*** doldurup veri iletimini yapmak, çevrim çalma ***(eyele steating)*** yöntemidir. Şekil 7'de bu iş için gerekli bağlantılar özetlenmektedir. Dolaysız bellek erişim devresindeki üç yazmaç önceden merkez işlem birimi tarafından doldurulur. Adres yazmacı iletilecek bilginin yaz/oku belleğin neresine yazılacağını ya da neresinden okunacağını sayaç iletilecek bilginin uzunluğunun kaç sözcük olacağını, konum yazmacı ise okuma mı yoksa yazma mı yapılacağını, dolaysız bellek erişiminin yapılıp yapılmadığını belirlemektedir. Şekil 8'de işlemin adımlan özetlenmiştir.

1) Çevre birimi DBEİS hattından dolaysız bellek erişimi isteyince DBE devresi ikind saat vuruşuna dek ENGEL imini gönderir.

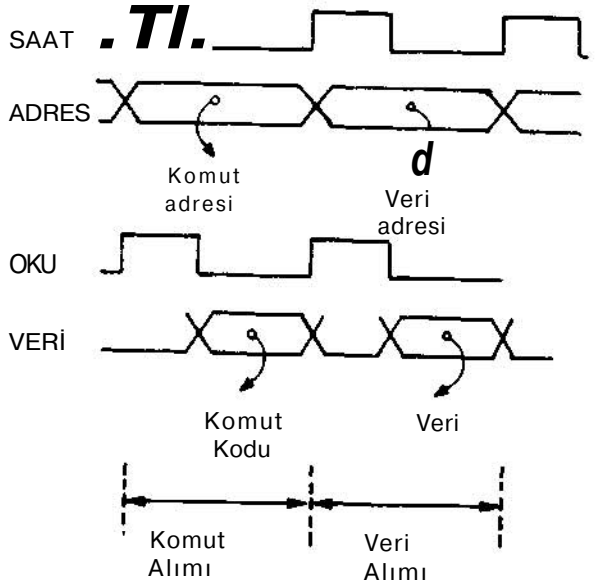
2) ENGEL, merkez işlem biriminde saatin bir vuruşunu engeller (MİBSAAT). Böylece Merkez işlem Birimi bir saat vuruşu süreyle doldurulmuş, yani bir "çevrimi çalınmış" olur.

3) DBE devresi adres yazmacındaki adresi ilk saat vuruşunda adres yoluna gönderir ve çevre birimine de DBEAL (Dolaysız bellek erişimi alındı) imini verir.
4) DBE'deki konum yazmacı DBEOY(dolaysız bellek erişimi oku/yaz) iletkenine " 1 " vererek çevre biriminin veri göndermesini, oku denetim yoluyla da yaz/oku belleğin okumasını sağlar.
5) Yeni bir saat vuruşunun gelmesi bu veri alışverişini durdurur. Bir sözcüktük verinin aktarılmasından sonra mikroişlemci programını yürütmeye devam eder. Çevre birimi, tamponunda gönderilecek yeni bilgi hazırlayınca yeniden DBEİS gönderip yeni bir iletimi başlatır. DBE devresi sayacında belirtilen miktardaki bilgi belleğe yerleştirilinceye dek birçok kez bu işlemler yinelenir, iletilecek tüm sözcükler iletilince DBE devresi bir kesme istemi göndererek merkez işlem biriminden yeniden komut (örneğin bu kez bir yazma işlemi) ister. Hep aynı tür iletimi yapmak için tasarımlanmış mikroişlemcili dizgelerde DBE devresinin adres yazmacı ve sayacını,, iletişimin başındaki durumuna getiren düzenek de yararlıdır.

Dolaysız bellek erişiminde bilginin DBE devresinden de geçmediğine dikkat edilirse birkaç çevre biriminin ko-



Şekil 8.



Şekil 9.

(aylıkla bir DBE tarafından denetlenebileceği gözlenir, örneğin 5 çevre birimini denetliyen 40 bacaklı bir DBE devresinin

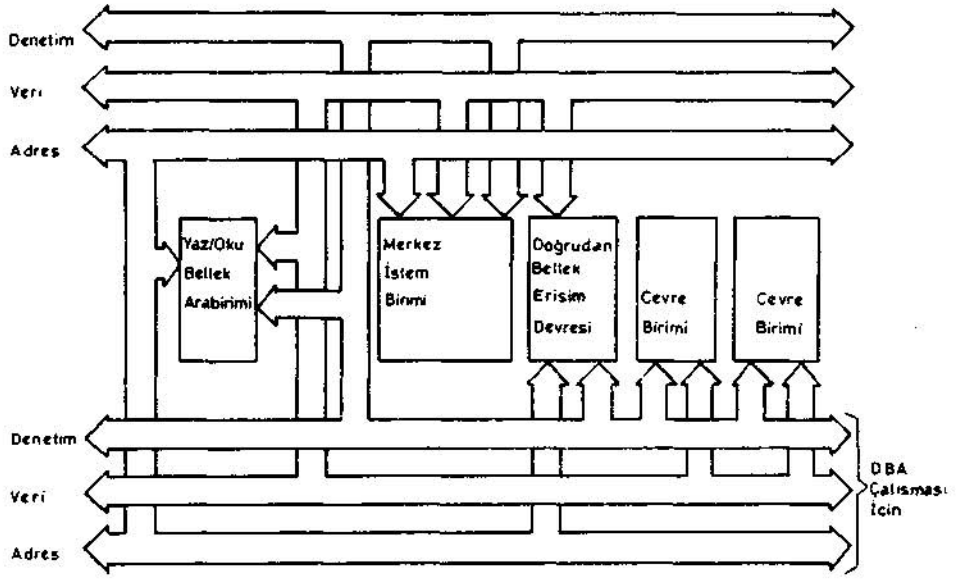
- 16 bacağı adres yoluna
- 3 bacağı kaynak-toprak
- 1 bacağı saat
- **8 bacağı veri yoiuna**
- 2 bacağı OKU ve YAZ iletkenlerine
- 1 bacağı ENGEL (dönem çalmak için)
- 2 bacağı KESİŞ, KESAL (DBE'ye son vermek için)
- 1 bacağı DBEAL
- 1 bacağı DBEOY
- S bacağı herbir çevre biriminden gelen DBEİS iletkenlerine bağlanabilir. Bu durumda kuşkusuz Adres, Konum yazmaçlarıyla sayaç sayılarını da 5'er tane olarak belirlemek gerekir.

Eşzamanlı Dolaysız Bellek Erişimi

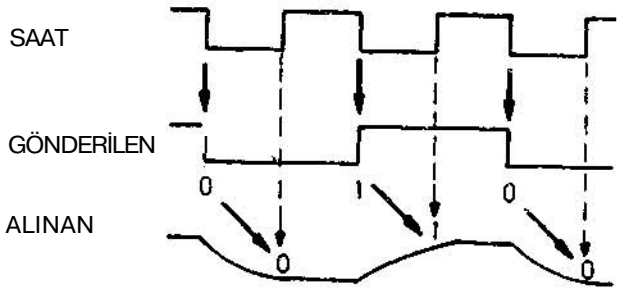
Dolaysız bellek erişiminin yalnızca merkez işlem biriminden dönem çalarak gerçekleşebileceğini düşünmek yanlış olur. Denetim, veri ve adres yollarından ikişer tane yaparak dönem çalmanın önüne geçilebilir, önce mikroişiemcili bir dizgede bellekten veri okuyan bir komutun Şekil 9'daki gibi işlem gördüğünü anımsayalım. Dikkat edilirse SAAT'ın yüksek olduğu taralı bölge içinde belleğe yazmak ya da okumak olanağı vardır. Şekil 10'da gösterildiği gibi donanım eklenerek bu başarılabilir. Yaz/oku bellekte farklı veri ve adres yolları ile ilişkilerin gerçekleştirilmesi için üç durumlu tampon ***(tristate buffer)*** adı verilen T bağlantılı tamponları gereklidir.

SERİ İLETİM

özellikle uzaktaki bir nokta ile iletişim yapan mikrobilgisayarların verileri sözcük bütünlüğü içinde koşut ***(pamüel)*** olarak alıp vermeleri olanaksızdır. Teker teker bitlerin ardarda gönderilmesinde ortaya çıkan ilk sorun alıcı ve vericinin bir eşzamanlama içinde çalışmasıdır. İletişim kanalında oluşabilecek bozulmaları gözönüne alıp örneğin iletim saatinin inen kenarında göndermek, çıkan kenarında almak tek saatli uygulamalarda sıkça başvurulan bir yöntemdir (Şekil 11). Bu arada iletim saatinin mikroişlemci dizge saatinden çok farklı olduğuna, örneğin iletim saatinin 110-9600 Hz arasenda olmasına karşılık, dizge saatinin 2 MHz dolayında olduğuna dikkat edilmelidir, iletim saat sıklığının iletim hızıyla aynı olması da



Şekil 10.



Sekil 11.

(Şekil 11) zorunlu değildir. Onaltı ya da altmışdört iletim saati vuruşunun bir bitlik bilgi îîetim süresi içinde yer aldığı dizgeler vardır ve gelen veri vuruşunun tam istenen noktasında örneklenmesini amaçlarlar. Çoğu kez vericinin ürettiği saat vuruşları alıcıya gönderilmez. Alıcı kendi saat vuruşlarını üretir ve eşzamanlı çalışması için verinin başında "eşzamanlama vuruşu" ya da "eşzamanlama örüntüsü" *(synciironizatior. pattern)* gönderilir.

iletişimde oluşacak yanılgıları sezmek ya da düzeltmek için eşlik *(parity)* bitleri üretmek "dönemsel fazlalık kodu" *(cyclic redundancy code)* kullanmak da çok kullanılan iletişim tekniklerindendir. En basit eşlik sınaması tek (ya da çift) eşlik sınaması adıyla bilinir. Sekiz bitlik bir sözcük iletilirken sonuna bir eşlik biti eklenip (XXXXXXXXP) 9 bitin içindeki "V'lerin sayısı tek (yada çift) yapılır. Alma yönünde de "V'lerin tek (ya da çift) olup olmadığı sınanır. Diğer yöntemde ise iletilecek bilgi belirli bir ikili *(binary)* çokterimliye bölünerek dönemsel fazlalık karakteri *(cyclic redundancy character)* elde edilir. Verinin ardında bu karakter de eklenip gönderilir. Alınan uçta da veri ardındaki karakterle çarpılıp çokterimli bulunmaya çalışılır.

Eşzamanlı veri iletiminde yalnızca verinin iletilmesi yeterli değildir, verici ile alıcı arasında bir "protokol" düzenine göre hazırlayıcı çalışmalar yapılır.

Örneğin IBM 2770 Bisync standardında

1) Verici bir istem gönderip alıcının hazır olup olmadığını sorar.
2) Alıcı hazır olduğunu bildirir.
3) Verici başladığını bildirip bir blok veri iletir. Ardına blok sonu bilgisini ve dönemsel fazlalık karakterini ekler.
4) iletimde yanılgı yoksa alıcı doğru blok aldığını iletir.
5) Verici yine başladığını bildirir, ardına veriyi ekler. Eğer bu, son veri bloku ise veri sonunu ve dönemsel fazlalık karakterini de ardından iletir.
6) Alıcı yanılgısız aldıysa alındı gönderir.
7) Verici bir iletim sonu karakteri gönderip susar.

Bu tür karşılıklı iletişim uygulamaları genel olarak anlaşma *(hand-shake)* adı altında toparlanırlar.

Eşzatnansız Seri İletim

İletimin eşzamanlamanın "sürekli" sağlanmadığı bir biçimde gerçekleştirilmesi de olasıdır. Karakterler arassnda vericinin çıkışı - çoğunlukla - yüksek konumda tutulur. Veri, iletilirken belirli çevreler içinde hazırlanır ve başına bir "0" biti sonuna iki " 1 " bit eklenip gönderilir: 0XXXXXXXX Pil (P eşlik biti).

Mikrobilgisayar ve minibilgisayar dizgelerinin önemli bir çevre aygıtı uzakyazıcı *(teletypewriter, TTY)* bir başlangıç biti, 7 veri biti, 1 eşlik biti ve 2 sonlandırrca bitinden oluşan 11 bitlik bir kod kullanır.

SERİ GİRİŞ /ÇIKIŞ İLETİŞİM YONGALARI

Mikroişlemci dizgelerinde kullanılan giriş/çıkış yongaları çoğunlukla 40 bacaktan çok daha küçük paketler içinde, örneğin 28 bacaklı birimler olarak piyasaya sürülmektedir. Eşzamanlı ve eşzamansız giriş/çıkış çoğunlukla aynı tümleşik devre ile gerçekleştirilebilir. Devrenin 8 bacağı mikroişlemcili dizgenin veri yolüna bağlanmaktadır. Kaynaklar, toprak ve dizge saati içinde 4 bacak ayrılır. Giriş/çıkış yongasının seçimi GÇSEÇ *(I/O Sekction, IOSEL)* bacağına yüksek gerilim uygulanması ile yapılır. Okuma ya da yazma işlemlerinden biri GÇOY *(I/O read/ vurite IORW)* iminin alçak ya da yüksek olmasıyla belirlenir. Seri veri alış (VA) *(receive Data, RD),* alınan saat (AS) *(Receive Clock, RC),* seri veri gönderme (VG) *(Transmit Data, TD),* gönderme saati (GS) *(Transmit clock, TC)* için de toplam 4 bacak ayrılır. Bazı birimlerde veri giriş ve çıkışı aynı bacaktan yapılmaktadır. Seri arabirimin üç önemli yazmacı tampon olarak kullanılır. Veri yolu tamponu, veri yolu ile koşut veri alışverişi yapar. Gönderme ve alma tamponları ise iletişim hattı ile seri ilişki, veri yolu tamponu ile koşut ilişki kurarlar.

Giriş/çıkış biriminin kendi dışıyla ilişkiler kurması, MODEM'le, Merkez işlem birimiyle gerekli uyum içinde çalışması için oldukça büyük sayıda (en az 8) denetim bağlantısı kurması gerekmektedir. Eşzamanlama vuruşları, gönderme tamponunun boş olduğunu belirten (bu nedenle merkez işlem biriminden veri isteyen) imler ya da alma tamponunun dolu olduğunu bildiren (mikroişlemci için kesme istemi olarak değerlendirilecek) im bu denetim bilgilerinin en önemlileridir.

KAYNAK

Adam Osborne, Introduction to Micro Computers, C.l, Basic Concepts, SYBEX Corp.