3 Giriş/Çıkış Organizasyonu (*I/O Organization*) ve Yol Erişimleri

Amaç, iç saklama birimleri (saklayıcılar, bellek) ile çevre birimler (tuş takımı, fare, modem, yazıcı, hard disk, ağ kartı) veri aktarımını sağlamaktır.

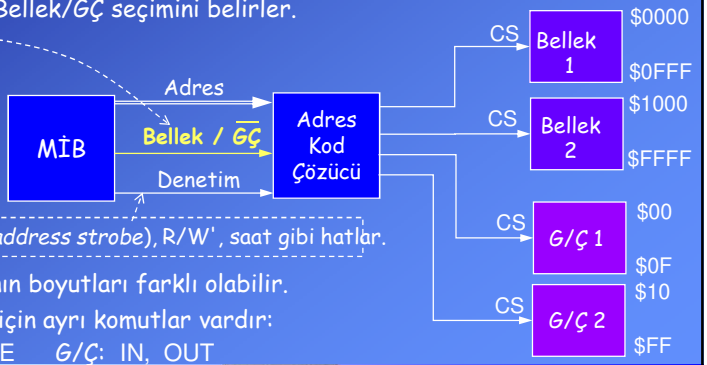
3.1 MİB – G/Ç Arabirimi Bağlantıları

Yalıtılmış G/Ç Haritası (*Isolated IO Map*) :

a) Bellek ve G/Ç birimleri için ayrı adres ve veri yolları olabilir.

b) Bellek ve G/Ç birimleri için ortak adres ve veri yolları olabilir.
Bir denetim hattı, Bellek/GÇ seçimini belirler.

Örnek:
Intel x86 işlemcileri



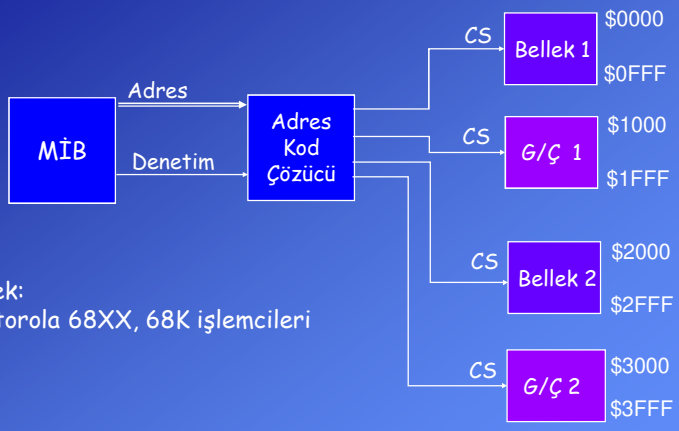
Geçerli adres (*address strobe*), R/W', saat gibi hatlar.

Bellek ve GÇ haritasının boyutları farklı olabilir.

Bellek erişimi ve G/Ç için ayrı komutlar vardır:
Bellek: LOAD, STORE G/Ç: IN, OUT

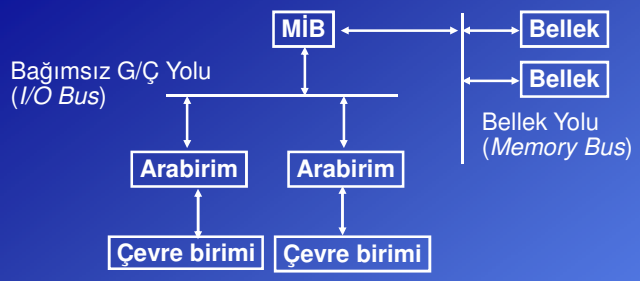
Bellek Haritalı G/Ç (*Memory Mapped IO*) :

- Aynı adres ve veri yolları hem bellek hem de G/Ç birimlerine erişimi için kullanılır.
- Belleklere ve G/Ç birimlerine aynı komutlar ile erişilir.
  MOVE, LOAD, STORE

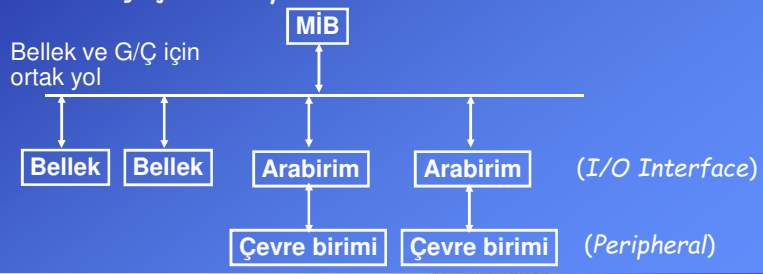


Önek:
Motorola 68XX, 68K işlemcileri

Bellek ve G/Ç için ayrı yollar olması:



Bellek ve G/Ç için ortak yol kullanılması:



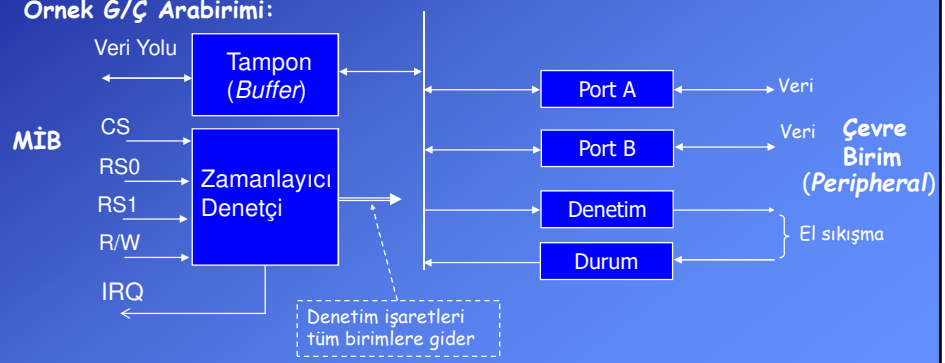
3.2 G/Ç Arabirimi (*I/O Interface*):

Çevrebirimler (*peripheral*), MİB yollarına G/Ç arabirimleri üzerinde bağlanır.

Arabirimin işlevleri:

- Hız farkını dengelemek
- Veri yapılarını, kodları dönüştürmek
- Hata düzeltme
- Çevre birimleri ile MİB arasında işaret dönüşümü. Manyetik, elektromekanik

Örnek G/Ç Arabirimi:



3.3 G/Ç Birimi ile Bellek Arasında Veri Aktarımı Yöntemleri:

1.Programlı Aktarım (Yoklamalı Çalışma "*Software Polling*"): MİB'in sorumlulukları:

a) Bir programla sürekli G/Ç ara biriminin bayraklarını gözlemlemek (meşgul, hazır)

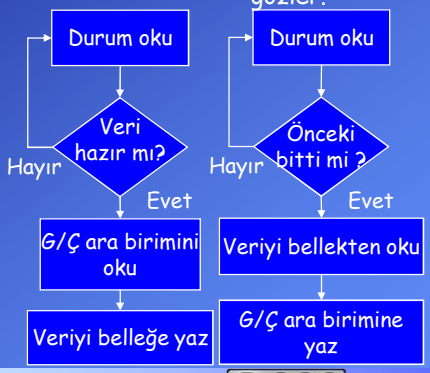
b) Veri aktarımını yapmak (G/Ç birimi ile saklayıcılar veya bellek arasında).

G/Ç ara biriminden okuma:

MİB bir programla G/Ç ara biriminin durumunu okur.

Eğer G/Ç arabirimi çevre birimden (*peripheral*) veriyi aldıysa "HAZIR" bayrağını çeker.

MİB G/Ç arabiriminde veriyi okur ve belleğe (saklayıcıya) yazar.



G/Ç ara birimine yazma:

MİB bellekten dışarıya veri göndermek istiyorsa G/Ç ara biriminin bayraklarını gözler.

Eğer G/Ç birimi bir önceki veriyi gönderdiyse "BİTTİ" bayrağını çekmiştir.

Bu durumda MİB bellekten veriyi okur ve G/Ç ara birimine yazar.

1. Programlı Aktarım (Yoklamalı Çalışma "*Software Polling*") devamı:

Olumsuz yanları:

Meşgul bekleme (*busy-waiting*)

G/Ç ile ilgili her iki işlem de MİB tarafından yapılır.

a) MİB G/Ç arabiriminin bayraklarını okumak için program koşturur.

Bu okuma sırasında MİB başka bir iş yapamaz (meşgul bekleme).

b) Veri aktarımı da MİB tarafından yapılır. Aktarılan veriler MİB'in üzerinden geçer.

Olumlu yanları:

Basittir; ek donanıma gerek duyulmaz.

Eğer MİB'in yapacak başka biri işi yoksa veya yapılan iş G/Ç bağımlı ise meşgul bekleme bir sorun oluşturmaz.

Bu tür sistemler için programlı aktarım uygun bir yöntemdir.

2. Kesmeli G/Ç (*Interrupt-Driven I/O*):

Kesmeli yöntemde, MİB G/Ç arabirimini hazır olduğunda kesme isteği üretecek şekilde koşullar.

Olumlu yanı:

MİB sürekli G/Ç ara biriminin bayraklarını gözlemek zorunda kalmaz, meşgul bekleme yoktur.

G/Ç arabirimi çevre birimden (*peripheral*) veri alırken (veya gönderirken) MİB diğer programları koşturabilir.

G/Ç arabirimi hazır olduğunda MİB'e kesme isteği gönderir.

MİB kesme isteği geldiğinde o andaki programı bırakır, veri aktarımını yapan kesme hizmet programını (KHP) çalıştırır ve tekrar kaldığı programa geri döner.

Bu yöntemde MİB bayrakların durumunu kontrol etme işini yapmaz ancak veriler hâlâ MİB üzerinden geçerek (programla) aktarılır.

Olumsuz yanı:

Kesme işlemlerinin ek yükleri vardır; geri dönüş adresini ve durum bilgisini saklamak, kesme hizmet programının adresini almak gibi (Bölüm 4).

Bu işlemler kesme hizmet programına her gidişte kesme çevriminde (*interrupt cycle*) yapılır. Geri dönülürken de geri dönüş adresi ve durum bilgisi okunur.

Çok sık veri aktarımı yapılan uygulamalar için kesmeli yöntem uygun değildir.

3. Doğrudan Bellek Erişimi (*Direct Memory Access* – DMA):

Hem programlı aktarımda hem de kesmeli iletimde veri aktarımını yapmak MİB'in görevidir.

Bu yöntemlerde verilerin okunup yazılması için MİB'de program koşturulur.

Doğrudan Bellek Erişimi (DMA) yönteminde ise ek bir donanım birimi olan doğrudan bellek erişimi denetçileri (***DMA controller***) (DMAC) kullanılır.

DMAC, MİB gibi sistem yolunu kullanan ve bellek erişimi yapabilen bir birimdir.

MİB veri aktarımına gerek duyduğunda denetçiyi koşullayarak hangi G/Ç ara birimi ile hangi bellek bölgesi arasında ne kadar veri aktarılacağını belirtir.

Bu koşulamadan sonra G/Ç işlemlerinden DMAC sorumludur.

G/Ç birimi hazır olduğunda DMA denetçisi sistem yolunu MİB'den alarak G/Ç ara birimi ile bellek arasındaki veri aktarımını yapar.

Bu sırada MİB kendi iç işlerini (bellek erişimi gerektirmeyen) sürdürür.

MİB ve DMA denetçisi sistem yolunu zaman paylaşımlı kullanırlar (biri kullanmadığında diğeri alır; DMA denetçisi önceliklidir).

DMA büyük miktarda, yoğun veri aktarılan uygulamalar için uygundur.

Ek donanım (DMAC) gerektirir.

Doğrudan bellek erişimi 5. bölümde ayrıntılı olarak açıklanacaktır.

Veri Aktarımı Yöntemleri Özeti:

Aşağıdaki tabloda hangi birimin G/Ç ara biriminin uygunluğunu gözlemlediği, hangisinin veri aktarımını gerçekleştirdiği gösterilmiştir.

Görev: / Yöntem:	G/Ç arabiriminin durumunu gözlemlemek	G/Ç arabirimi ile bellek arasında veri aktarmak
Programlı G/Ç	MİB (Program)	MİB (Program)
Kesmeli G/Ç	Kesme mekanizması	MİB (KHP)
DMA	DMAC	DMAC

3.4 Asenkron Veri Aktarımı:

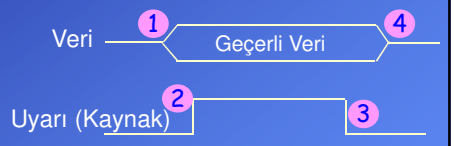
Sorunlar:

a) Kaynak (gönderen) veriyi gönderdi mi (Veri yolundaki veri geçerli mi)?

b) Varış (alıcı) veriyi aldı mı (alıcı meşgul mü, uygun mu)?

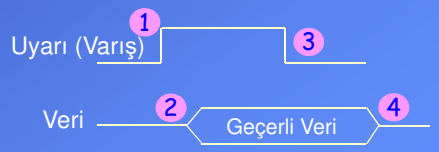
1. Uyarmalı İletim (*Strobe Control*):

a) Kaynak başlatmalı uyarma:



Geçerli verinin ne kadar süre yolda kalacağı varış birimine göre önceden belirlenir. Gönderen, verinin alıcıya ulaşıp ulaşmadığını bilemez.

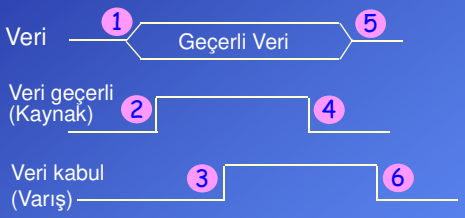
b) Varış başlatmalı uyarma :



Verinin alıcı tarafından yoldan ne zaman örnekleneceği (alınacağı) kaynak birimin hızına göre önceden belirlenir.
Alıcı, gönderenin veriyi gerçekten yola koyup koymadığını bilemez.

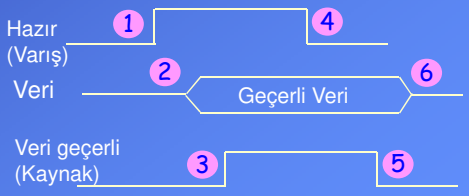
2. El Sıkışma (*Handshaking*):

a) Kaynak başlatmalı:



Kaynak, "Veri kabul" işareti gelinceye kadar veriyi yolda tutar.
Arıza durumlarında sonsuz beklemeyi önlemek için zamanlayıcı kullanılır.
Buna **zaman aşımı (*time-out*)** mekanizması denir.

b) Varış başlatmalı:



Varış, "Veri geçerli" gelinceye kadar bekler.
Zaman aşımı (*time-out*) mekanizması burada da gereklidir.

3.5 MİB – Bellek (veya G/Ç Arabirimi) Arasında Veri Aktarımı:

Merkezi işlem birimleri de bellek erişiminde senkron ya da asenkron veri aktarım yöntemlerinden birini kullanırlar.

3.5.1 Senkron ve uyarmalı yol erişimi

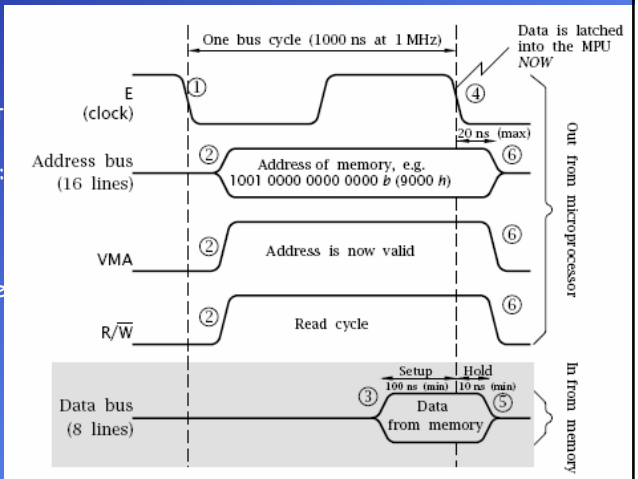
Örneğin, MC 6802 mikroişlemcisi bellek erişiminde **uyarmalı (*strobe*)** yöntemi kullanır ve erişim saat işareti (E) ile senkronlanır.

VMA (*Valid Memory Address*): Geçerli adres olduğunu belirtir ve erişimi başlatır (uyarma işareti).

Saat işareti 1'den 0'a indiğinde yol çevrimi tamamlanır.

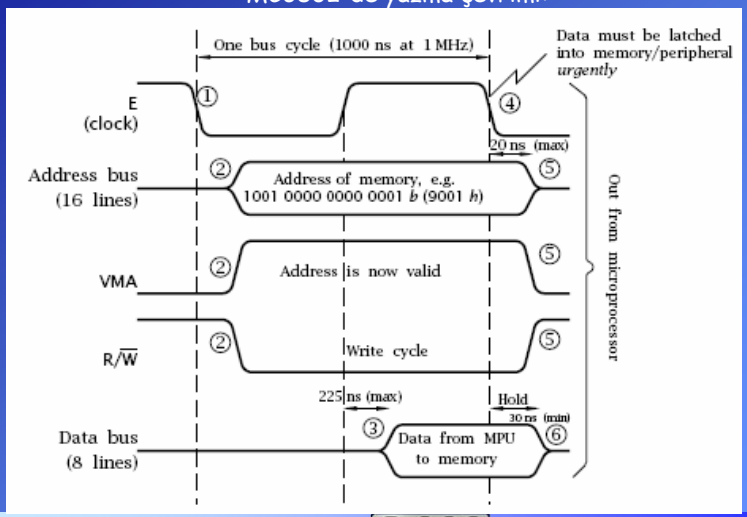
Yandaki şekilde 6802'nin bellekten okuma işleminin zamanlama diyagramı gösterilmiştir.

MC6802'de okuma çevrimi:



Benzer şekilde yazma erişiminde de çevrim VMA'nın etkin olmasıyla başlar.
Adres kod çözücü üzerinden ilgili bellek birimine seçme işareti (*chip select*) gider.
Saat işareti 1'den 0'a indiğinde yol çevrimi tamamlanır.

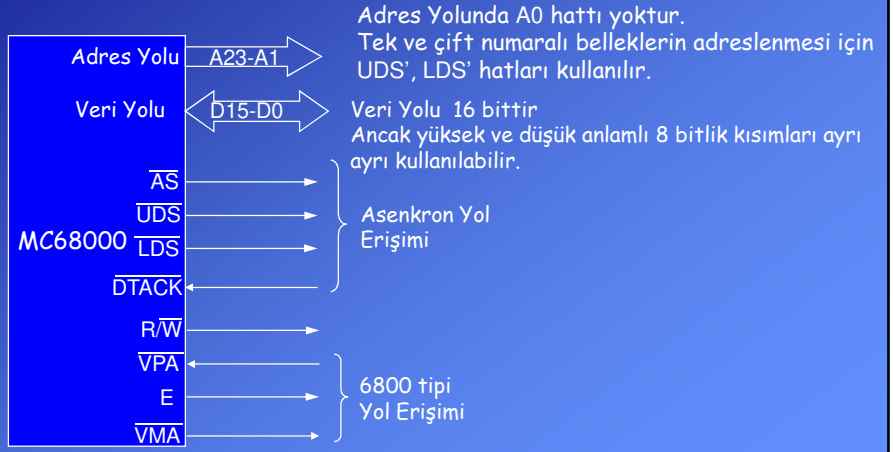
MC6802'de yazma çevrimi:



3.5.2 Asenkron ve El sıkışmalı Yol Çevrimi

Örneğin, MC68000 belleğe (ve G/Ç arabirimlerine) **asenkron el sıkışmalı** yöntemle erişir.

Eğer istenirse 6800 ailesinin elemanları gibi (uyarmalı ve senkron) olarak da yolu kullanabilir.



MC68000'in bellek erişiminde kullanılan denetim işaretleri

- AS' (*Address Strobe*): İşlemci bu işareti etkin (sıfır) yaparak adres yolunda geçerli bir adres olduğunu gösterir

  Bu işaret yol çevrimini başlatır. İlk el sıkışma işaretidir.

- UDS' (*Upper Data Strobe*) ve LDS' (*Lower Data Strobe*): Erişilen verinin boyutunu (8/16 bit) ve adresin tek/çift olmasını belirtirler.

  Word (16 bit): Her ikisi de etkin (sıfır)

  Byte (8 bit, tek adres): LDS' etkin, D0-D7 kullanılır

  Byte (8 bit, çift adres): UDS' etkin, D8-D15 kullanılır

- DTACK' (*Data Transfer Acknowledge*): 68000'in El sıkışma girişi

  Adreslenen birim (bellek veya G/Ç) tarafından etkin (lojik 0) yapılırsa adreslenen biriminin veriyi aldığı/gönderdiği anlaşılır.

  Yol çevrimi tamamlanmış olur. Son el sıkışma işaretidir.

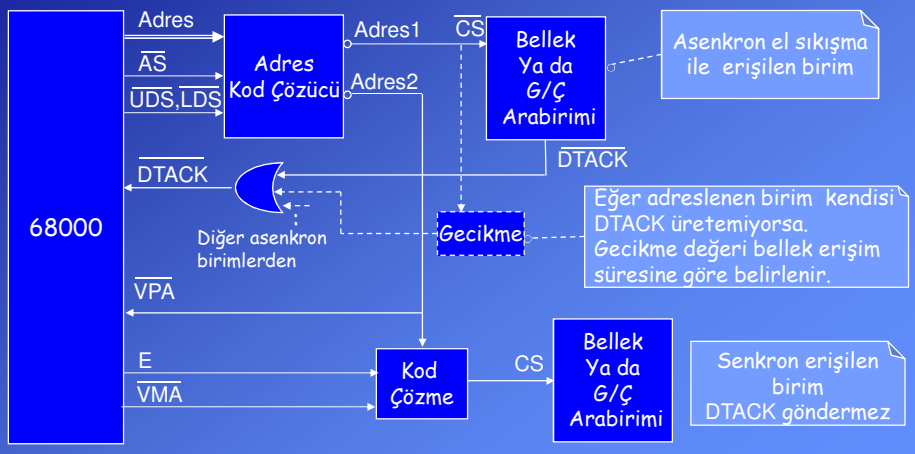
- VPA' (*Valid Peripheral Address*): Bu giriş, yol çevrimi başladığında (AS' etkin) dışarıdan etkin (lojik 0) yapılırsa adreslenen birimin ancak 6800 tipi yol erişimi (uyarmalı) yapabildiği 68000'e bildirilmiş olur.

  Bu durumda ilgili birime VMA ve E işaretleri kullanılarak erişilir.

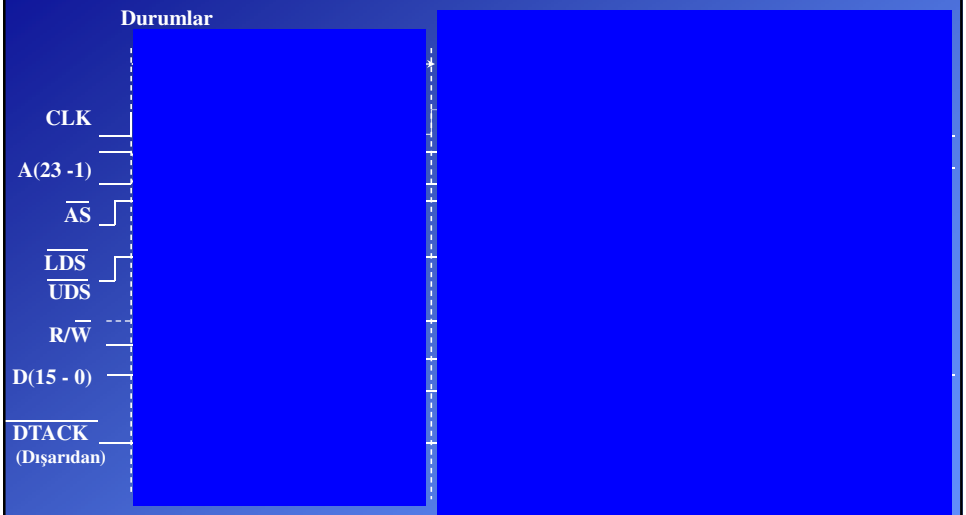
MC68000 Bellek (G/Ç Arabirimi) Bağlantısı

MC68000 belleğe (ve G/Ç arabirimlerine) asenkron el sıkışmalı yöntemle erişir.

Eğer istenirse 6800 ailesinin elemanları gibi (uyarmalı ve senkron) olarak da yolu kullanabilir.

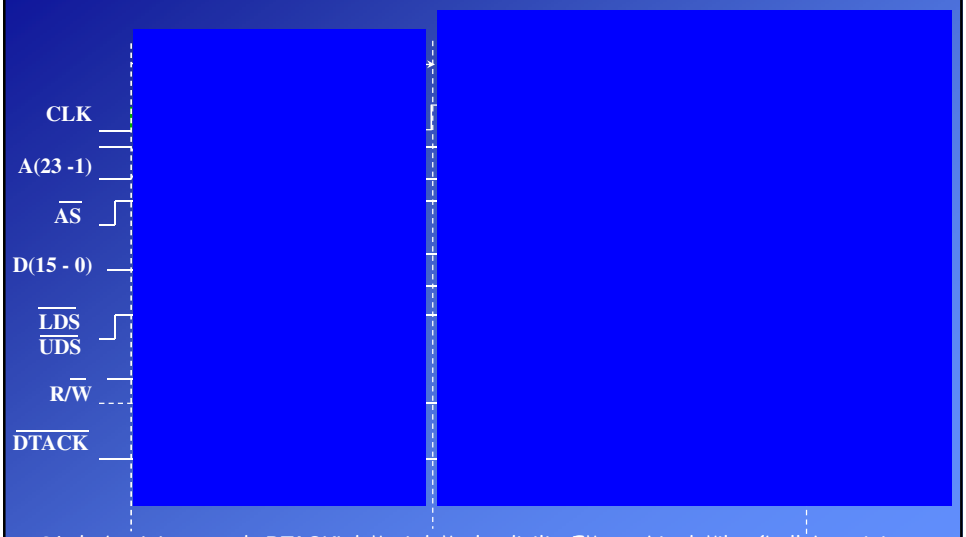


MC68000 Asenkron Yol Erişimi Okuma Zamanlaması



S6 durumunun sonundaki (S7'nin başında) inen kenarda değer yoldan okunur.

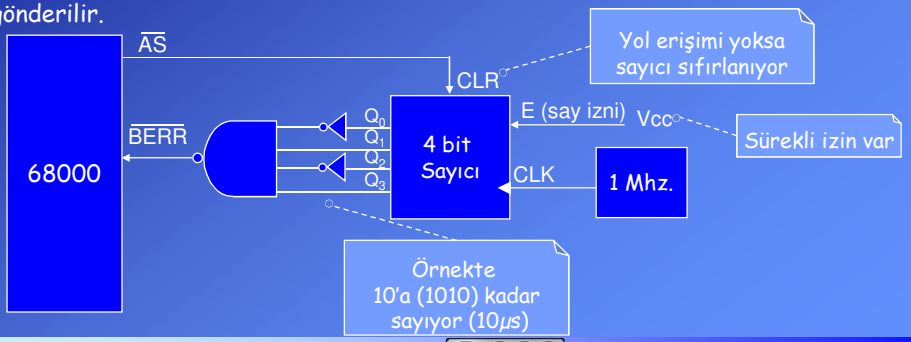
MC68000 Asenkron Yol Erişimi Yazma Zamanlaması



S6 darbesinin sonunda DTACK' değeri değerlendirilir. Eğer etkin değilse (bellek veriyi almamışsa) bekleme durumları (W) eklenerek yol çevrimi uzatılır.

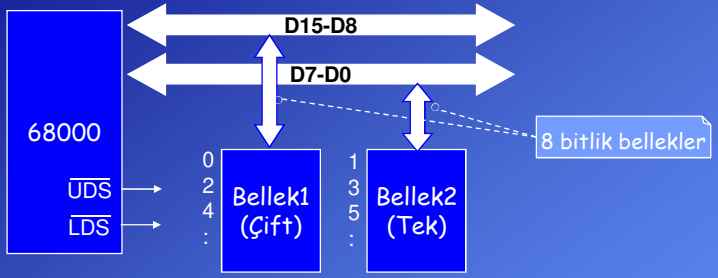
Sonsuz Beklemeyi Önleme

Yol erişimindeki hatalara karşı önlem alabilmek için MC68000'in **BERR'** (*Bus Error*) girişi kullanılır.
Bu giriş etkin (lojik 0) olduğunda 68000 o andaki yol çevrimini keser, yığına o andaki durumu ile ilgili bilgiler (son çıkarılan adres, yürütülmekte olan komut vb.) yazar ve belirli bir hizmet programına gider.
BERR' ile ilgili işlemler "Sıra Dışı Durumlar" bölümünde açıklanacaktır.
Sonsuz beklemeyi önlemek için 68000'in BERR' girişine aşağıdaki gibi bir sayıcı bağlanabilir.
Eğer yol çevrimi beklenenden uzun sürerse (AS' etkin kalırsa) işlemciye BERR' gönderilir.



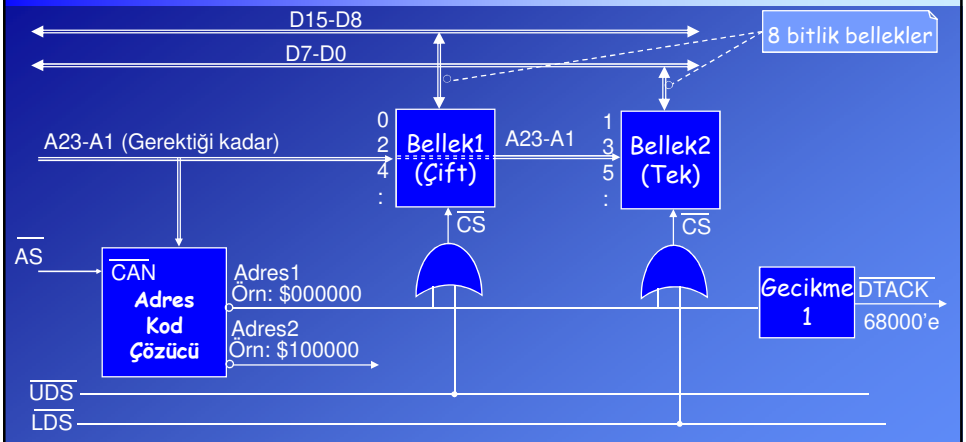
MC68000'de 8/16 bitlik (tek/çift adreslere) erişim

MC68000 mikroişlemcisinin veri yolunun genişliği 16 bittir.
Komutlar 8, 16 ya da 32 bitlik bellek erişimlerini gerektirmektedir.
Bu nedenle 16 bitlik veri yoluna 8 bitlik bellekler paralel olarak bağlanır.



Hangi belleğin adresleneceğini belirtmek için MC68000 mikroişlemcisinin iki adet çıkışı bulunur: **UDS'** (*Upper Data Strobe*) ve **LDS'** (*Lower Data Strobe*).
Adres yolunda A0 hattı yoktur.

UDS	LDS	D15-D8	D7-D0	Anlamı:	
H	H	---	---	Bellek erişimi yok	
H	L	---	Veri	Tek numaralı adrese byte erişimi	A0 = 1
L	H	Veri	---	Çift numaralı adrese byte erişimi	A0 = 0
L	L	Veri	Veri	Çift numaralı adrese word erişimi	A0 = 0



Örnek:

Adres: (A_0 dışarı çıkmaz)

	$A_{23}A_{22}$ ..... A_1 A_0	UDS	LDS
MOVE.B ($000000),D1	0 0 ..... 0 0	0	1
MOVE.B ($000001),D1	0 0 ..... 0 1	1	0
MOVE.W ($000000),D1	0 0 ..... 0 0	0	0
MOVE.W ($000001),D1	Adres hatası. Sıra dışı durum (*exception*) oluşur.		

Örnek:

Yandaki örnekte MC68000 mikroişlemcisine, bir adet seri iletişim arabirimi (MC6850), toplam 4Kx8'lik salt oku bellek (iki adet 2716) ve toplam 4Kx8'lik yaz oku bellek (iki adet 4016) bağlanmıştır.

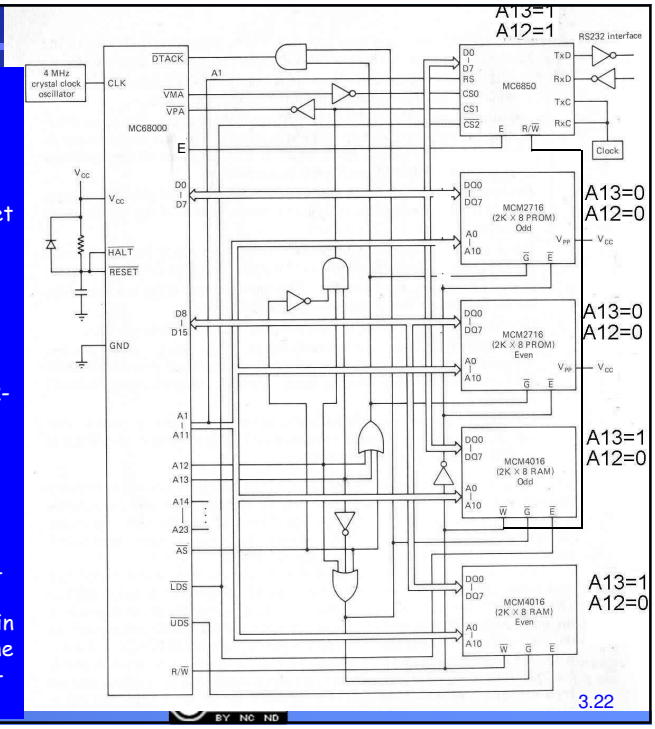
Hangi birimin adresleneceği A13 ve A12 adres hatları ile belirlenmektedir.

Seri iletişim arabirimi (MC6850), mikroişlemciyle senkron olarak iletişim kurmaktadır (VMA ve E kullanılıyor).

Not: Karşı taraftaki seri iletişim birimi ile **asenkron** olarak haberleşir.

Bellekler ise mikroişlemci ile asenkron olarak haberleşmektedirler (AS ve DTACK kullanılır).

ROM'lara yazma yapılmadığı için bu elemanların kod çözücülerine UDS/LDS hatlarının bağlanması sorun olmaz.



MC68000'de İşlev (Durum) Kodları:

MC68000 mikroişlemcisinin durumunu gösteren 3 bitlik bir çıkışı vardır:

Function Codes Outputs: FC2, FC1, FC0.

Bu çıkışlar her yol çevriminde (AS' etkin olduğunda) geçerli değer alırlar ve mikroişlemcinin durumunu dışarıya gönderirler.

Kullanıcı ve yönetici konumları 4.5.1 Ayrıcalık Konumları (*Privilege Modes*) bölümünde açıklanmıştır.

FC2	FC1	FC0	Anlamı:
0	0	0	Tanımsız (Rezerve)
0	0	1	Kullanıcı Konumu, Veri erişimi (*User Data*)
0	1	0	Kullanıcı Konumu, Program erişimi (*User Program*)
0	1	1	Tanımsız (Rezerve)
1	0	0	Tanımsız (Rezerve)
1	0	1	Yönetici Konumu, Veri erişimi (*Supervisor Data*)
1	1	0	Yönetici Konumu, Program erişimi (*Supervisor Program*)
1	1	1	Kesme Kabul (*Interrupt Acknowledge*)

Bu çıkışları da adres kod çözmede uygun şekilde kullanarak değişik işlemler gerçekleştirmek mümkündür. Örneğin;

- Belli elemanlara ve adres bölgelerine sadece yönetici konumunda erişilmesine izin verilir,
- Program ve veri belleği ayrılabilir.

Örnek: Aynı adrese program ve veri belleklerinin yerleştirilmesi

Aşağıdaki örnekte MC68000'nin FC0 çıkışı kullanılarak aynı adrese farklı iki bellek yerleştirilmiştir.
Bellek gruplarından birincisi FC0=0 olduğunda (program erişimi),
ikincisi ise FC0=1 olduğunda (veri erişimi) seçilmektedir.