

Bölüm 2: 8086 Pinler

Mikroişlemciler

40 pinli DIP paket

Adres/veri çoklama

ALE ve tampon/kilit

BHE' ve bellek bankaları

Okuma/yazma döngüleri

Durum pinleri

Kontrol pinleri

Minimum / maksimum mod

8284A ve 8288

Kesme pinleri ve vektör tablosu

Sercan KÜLCÜ · Giresun Üniversitesi · Elektrik-Elektronik Mühendisliği

Öğrenme Çıktıları

Pin işlevleri

8086'nın 40 pinini gruplarına ayırmak;
aktif düşük/yüksek sinyalleri okumak.

Çoklama

Adres ve verinin aynı pinlerden nasıl
geçtiğini, ALE ve adres kilitlerinin rolünü
açıklamak.

Yol döngüleri

T1–T4 durumlarında okuma ve yazma
döngüsünü, bekleme durumunu (Tw)
zamanlama diyagramından izlemek.

Sistem kurulumu

Minimum/maksimum mod farkını, 8284A,
8288 ve kesme mekanizmasını tanımak.

8086 Paketi

- Intel 8086, 16 bitlik **HMOS** mikroişlemcidir.
- **40 pinli DIP** (dual in-line package) tümleşik devre olarak üretilir.
- Tek **+5 V DC** besleme ile çalışır (pin 40 = V_{CC} , pin 1 ve 20 = GND).
- **20 hatlı** adres yolu (A0–A19) ve **16 hatlı** veri yolu (D0–D15).
- Adres yolu **çoklamalı** (multiplexed) çalışır: bir pin farklı zamanlarda farklı amaçla kullanılır.
 - 16 düşük adres hattı (A0–A15) veri ile çoklanır → **AD0–AD15**.
 - 4 yüksek adres hattı (A16–A19) durum bilgisi ile çoklanır → **A16/S3 ... A19/S6**.

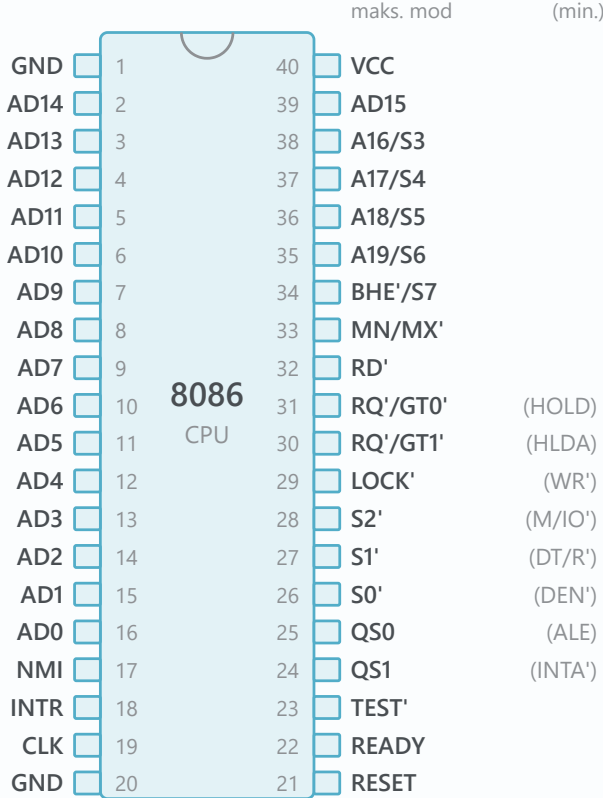
Gösterim

Adının sonunda **'** (veri sayfasında üst çizgi) olan sinyal **aktif düşüktür**: işlevi 0 V'ta yapar. Ör. **RD'** = 0 → okuma.

MN/MX' gibi adlarda: 1 iken MN, 0 iken MX anlamı geçerlidir.

Yön: G = giriş, Ç = çıkış, G/Ç = çift yönlü.

Pin Diyagramı (Etkileşimli)



maks. mod

min. mod

8086

40 pin · maksimum mod

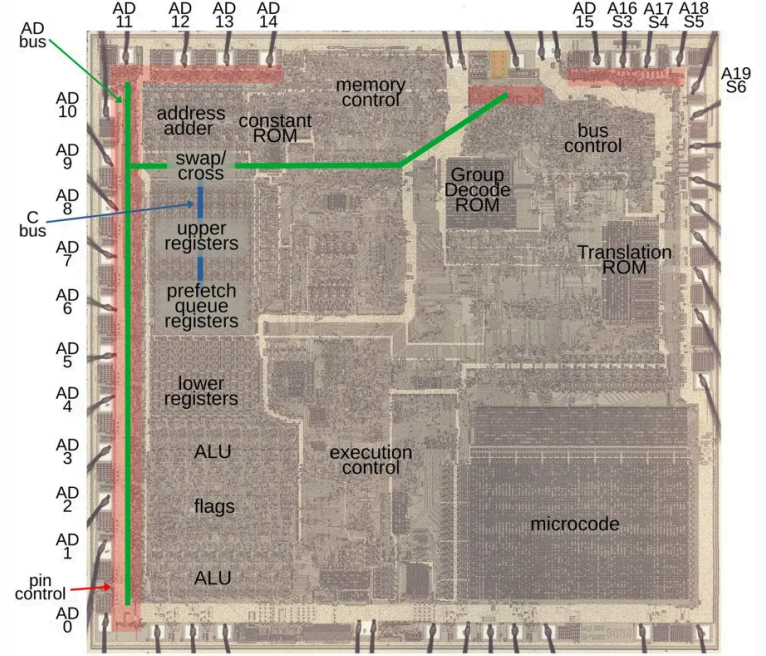
Pin 24–31'in adı moda göre değişir. Mod düğmeleriyle karşılaştır.

Bir pine tıkla (ya da Tab ile seç).

GENEL BAKIŞ

Yonganın İçi: Kalıp Fotoğrafı

- Silikon kalıp (die), paketin ortasında yer alır; kenarlardaki **bağlantı pedleri** ince tellerle 40 pine bağlanır.
- Sol kenardaki **AD yolu** pedleri, adres/veri pinlerine giden hatlardır.
- Sol tarafta 16 bitlik yazmaçlar, ALU ve önden getirme kuyruğu yazmaçları; sağ altta komutları yürüten **mikrokod ROM'u** görülür.
- Yaklaşık 29 000 transistör, 3 µm HMOS süreci.



Fotoğraf: Ken Shirriff (righto.com), işlevsel bloklar işaretli.

Neden Karmaşık Pinler?

Mikroişlemciler bellek ve G/Ç aygıtlarına veri yolu ve adres yolu üzerinden bağlanır. 8086'da bu bağlantı iki nedenle karmaşıktır:

1. 40 pinli paket yetersiz

- 20 adres + 16 veri + ~20 kontrol + güç/saat $\approx$ 60'tan fazla sinyal gerekir.
- Çözüm: adres, veri ve durum bilgisi **aynı pinler üzerinden, farklı zamanlarda** iletilir (zaman paylaşımli çoklama).

2. 20 bit adres, 16 bit veri

- Kesim yazmaçlarıyla üretilen adres 20 bittir; veri yolu yalnızca 16 bittir.
- Fazladan 4 adres biti (A16–A19) kendi pinlerini durum sinyalleriyle (S3–S6) paylaşır.



Bedeli: dışarıda adresi **kilitleyen** (latch) ve veriyi **tamponlayan** ek devreler gerekir; ayrıca her bellek erişimi en az 4 saat turu sürer.

Pin Grupları

Grup	Pinler	Not
Güç	V _{CC} (40), GND (1, 20)	+5 V ± %10
Saat ve sıfırlama	CLK (19), RESET (21), READY (22)	8284A saat üreticinden gelir
Adres/veri	AD0–AD15 (2–16, 39)	T1'de adres, sonra veri
Adres/durum	A16/S3–A19/S6 (38–35), BHE'/S7 (34)	T1'de adres, sonra durum
Kesme	NMI (17), INTR (18), INTA' (24, min)	donanım kesmeleri
Ortak kontrol	RD' (32), TEST' (23), MN/MX' (33)	her iki modda aynı
Moda bağlı	24–31	min: INTA', ALE, DEN', DT/R', M/IO', WR', HLDA, HOLD maks: QS1, QS0, S0', S1', S2', LOCK', RQ'/GT1', RQ'/GT0'

Minimum ve Maksimum Mod

- **MN/MX' = 1 (V_{CC})** → minimum mod: tek işlemcili küçük sistem; kontrol sinyallerini 8086 kendisi üretir.
- **MN/MX' = 0 (GND)** → maksimum mod: çok işlemcili/yardımcı işlemcili (8087) sistem; kontrol sinyallerini **8288 yol denetleyicisi** üretir, açılan pinler durum ve yol arbitrajı için kullanılır.
- Mod donanımıyla, pin sabit bağlanarak seçilir; çalışırken değişmez.

Pin	Minimum	Maksimum
24	INTA'	QS1
25	ALE	QS0
26	DEN'	S0'
27	DT/R'	S1'
28	M/IO'	S2'
29	WR'	LOCK'
30	HLDA	RQ'/GT1'
31	HOLD	RQ'/GT0'

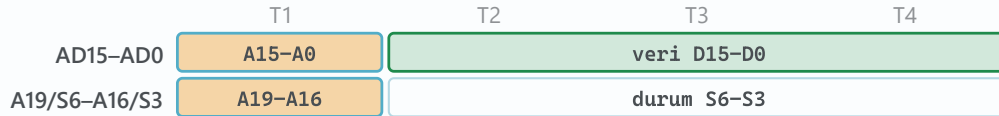
Adres/Veri Yolu Pinleri

AD0–AD15 · Adres/Veri yolu

- Düşük 16 adres hattıdır; veri hattı ile çoklanır.
- Bellek adresi taşırken **A** ile gösterilir: A0–A15 (T1).
- Veri taşırken **D** ile: D0–D7, D8–D15 ya da D0–D15 (T2–T4).
- Çift yönlüdür (yazmada çıkış, okumada giriş); gerekirse yüzer (float, yüksek empedans) duruma geçer.

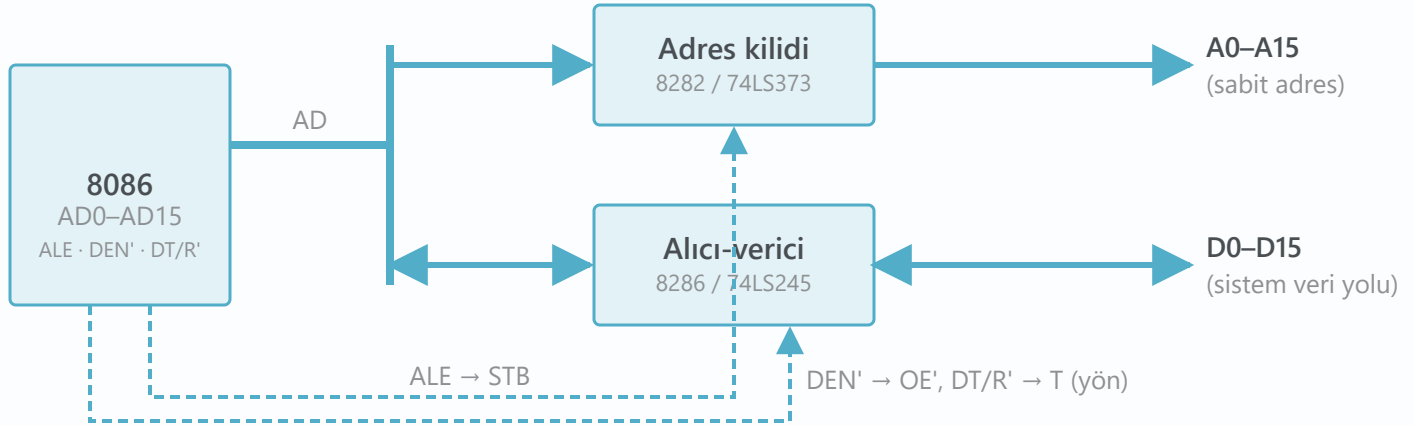
A16/S3 – A19/S6 · Adres/Durum

- Yüksek 4 adres hattıdır; durum sinyalleri ile çoklanır.
- T1'de A16–A19 (bellek erişiminde), T2–T4'te S3–S6.
- G/Ç erişiminde T1 sırasında 0'dır (G/Ç adresi 16 bit).
- S3, S4: hangi kesimin kullanıldığı; S5: IF bayrağı; S6: her zaman 0.



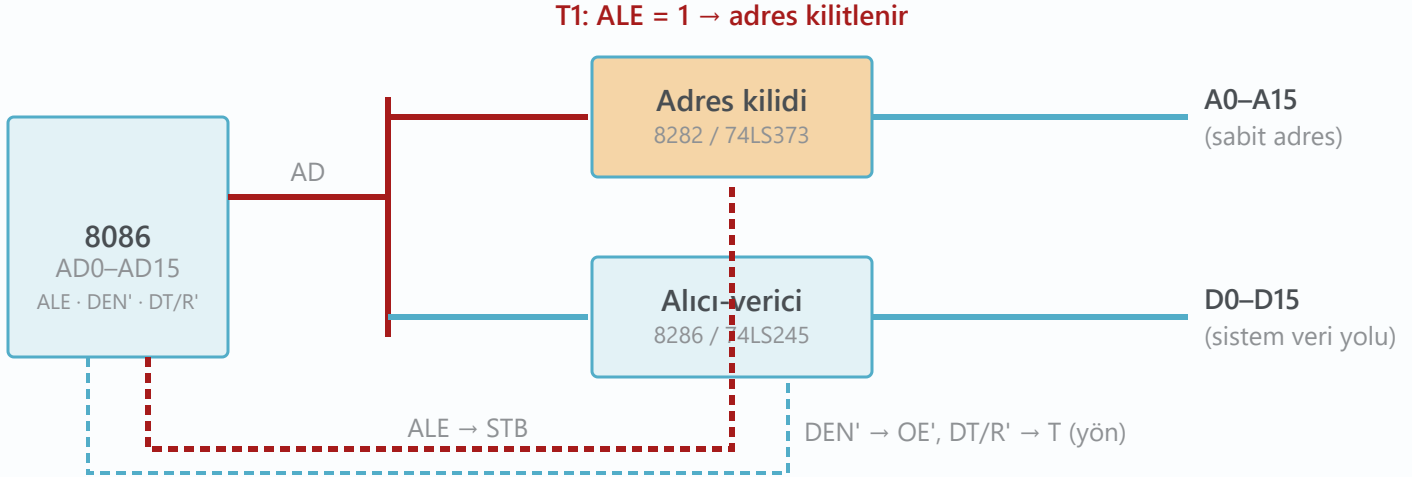
BHE'/S7 (pin 34): T1'de veri yolunun üst yarısının (D8–D15) kullanılacağını bildirir; sonra S7 durum bitine döner. **RD'** (pin 32): okuma darbesi.

Çoklamayı Çözmek: ALE, Kilit ve Tampon



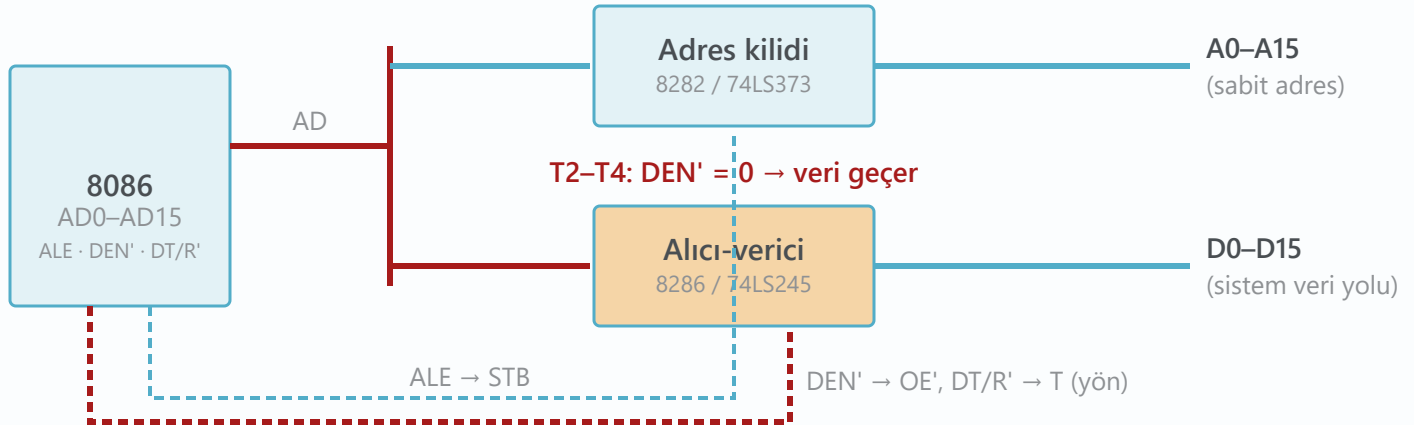
AD0-AD15 pinleri hem adres hem veri taşır. Sistemin geri kalanı ayırık ve kararlı bir adres yolu ile veri yolu ister.

Çoklamayı Çözmek: ALE, Kilit ve Tampon



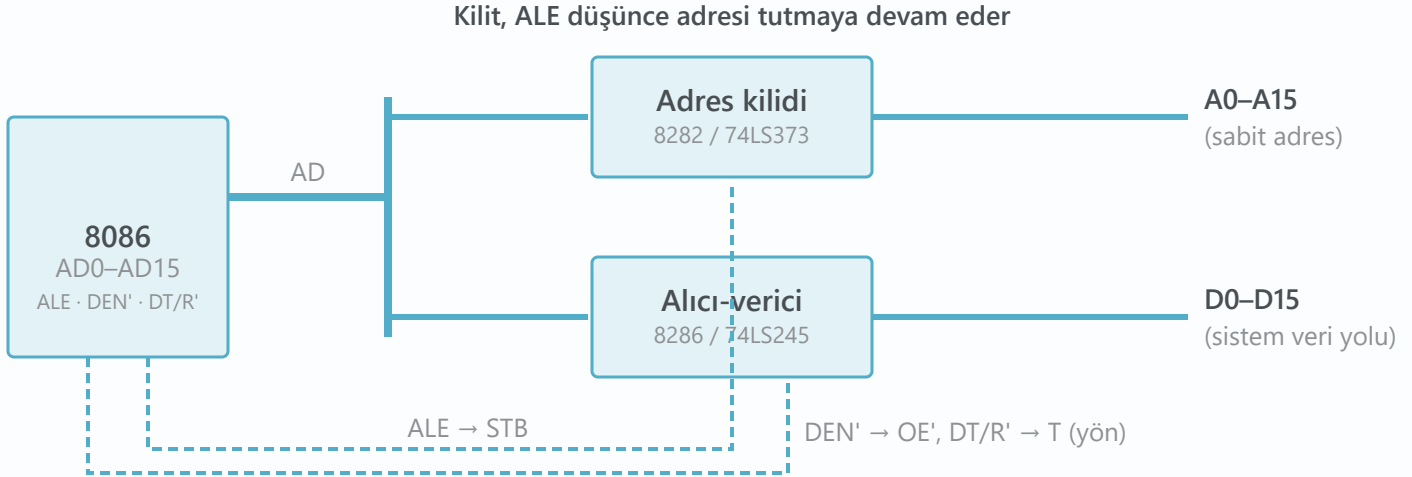
T1: 8086 adresi AD pinlerine koyar ve **ALE**'yi 1 yapar. ALE'nin düşen kenarında 8282 kilidi adresi yakalar.

Çoklamayı Çözmek: ALE, Kilit ve Tampon



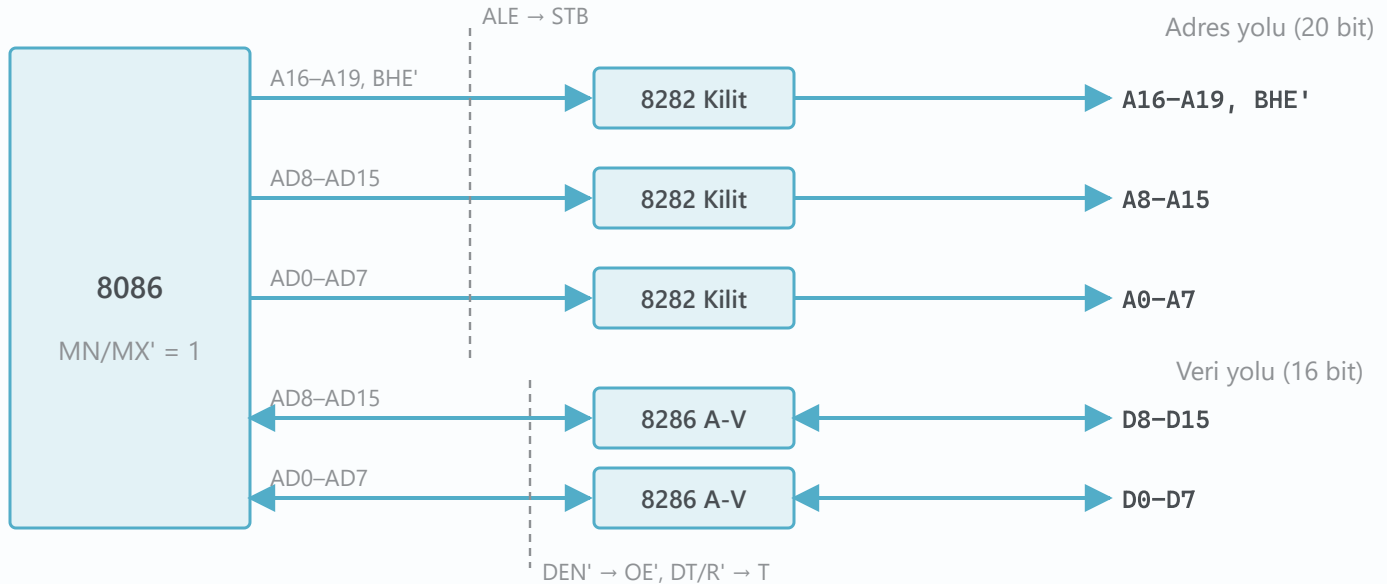
T2-T4: aynı pinler artık veri taşır. **DEN'** = 0 alıcı-vericiyi açar, **DT/R'** yönü seçer (1: CPU → bellek, 0: bellek → CPU).

Çoklamayı Çözmek: ALE, Kilit ve Tampon



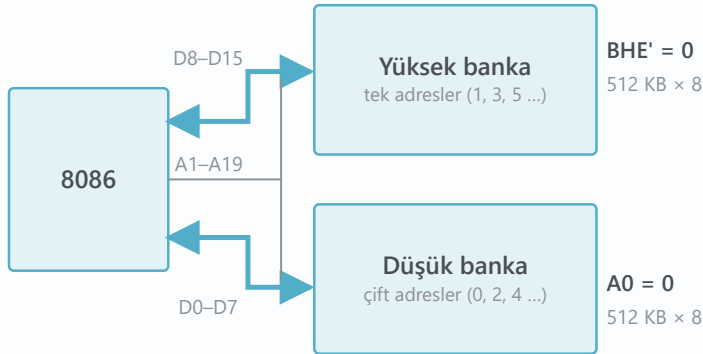
Kilit adresi tüm döngü boyunca bellek ve G/Ç'ye sabit sunar. A16-A19/BHE' için de ayrı bir kilit kullanılır (toplam 3 adet 8282).

Tamponlu ve Kilitli Sistem



Tamponlar (buffer) hem çoklamayı çözer hem de **sürme gücü** sağlar: 8086'nın pinleri ancak sınırlı sayıda yükü sürebilir; büyük sistemlerde tüm yollar tamponlanır.

BHE' ve A0: Bellek Bankaları



BHE'	A0	Aktarım
0	0	tüm sözcük, D0-D15 (çift adres)
0	1	yüksek bayt, D8-D15 (tek adres)
1	0	düşük bayt, D0-D7 (çift adres)
1	1	yok

1 MB bellek iki 512 KB'lık bankaya bölünür. **Tek adresteki** bir sözcük (ör. 20101h) tek döngüde aktarılamaz: 8086 bunu otomatik olarak **iki yol döngüsüne** böler. Verileri çift adrese hizalamak (EVEN direktifi) hız kazandırır.

Kod İzleme: Hizalı ve Hizasız Erişim

AX ← 2000h. Yalnızca komut getirme döngüleri; veri erişimi yok.

Adım 1 / 10

► **mov ax, 2000h**

```
2 mov ds, ax
3 mov ax, 1234h
4 mov [0100h], ax ; çift adres
5 mov ax, 0ABCDh
6 mov [0101h], ax ; tek adres
7 mov al, [0100h] ; bayt
8 mov bx, [0101h] ; tek adres
```

DEĞİŞKENLER

AX = **2000h**

DS = ?

BX = ?

döngü = -

FA = -

BHE' = -

A0 = -

D15-D8 = -

D7-D0 = -

RD'/WR' = **1/1**

[20100h] = ??

[20101h] = ??

[20102h] = ??

Kod İzleme: Hizalı ve Hizasız Erişim

DS ← 2000h: veri kesimi 20000h'de başlar.

Adım 2 / 10

```
1 mov ax, 2000h
▶ mov ds, ax
3 mov ax, 1234h
4 mov [0100h], ax    ; çift adres
5 mov ax, 0ABCDh
6 mov [0101h], ax    ; tek adres
7 mov al, [0100h]    ; bayt
8 mov bx, [0101h]    ; tek adres
```

DEĞİŞKENLER

AX = 2000h

DS = 2000h

BX = ?

döngü = -

FA = -

BHE' = -

A0 = -

D15-D8 = -

D7-D0 = -

RD' / WR' = 1/1

[20100h] = ??

[20101h] = ??

[20102h] = ??

Kod İzleme: Hizalı ve Hizasız Erişim

AX ← 1234h (AH=12h, AL=34h).

Adım 3 / 10

```
1 mov ax, 2000h
2 mov ds, ax
▶ mov ax, 1234h
4 mov [0100h], ax ; çift adres
5 mov ax, 0ABCDh
6 mov [0101h], ax ; tek adres
7 mov al, [0100h] ; bayt
8 mov bx, [0101h] ; tek adres
```

DEĞİŞKENLER

AX = 1234h

DS = 2000h

BX = ?

döngü = -

FA = -

BHE' = -

A0 = -

D15-D8 = -

D7-D0 = -

RD'/WR' = 1/1

[20100h] = ??

[20101h] = ??

[20102h] = ??

Kod İzleme: Hizalı ve Hizasız Erişim

FA = 20100h **çift**: BHE'=0, A0=0 → iki banka birden. Sözcük **tek döngüde** yazılır; AL çift adrese (34h), AH tek adrese (12h).

Adım 4 / 10

```
1 mov ax, 2000h
2 mov ds, ax
3 mov ax, 1234h
▶ mov [0100h], ax ; çift adres
5 mov ax, 0ABCDh
6 mov [0101h], ax ; tek adres
7 mov al, [0100h] ; bayt
8 mov bx, [0101h] ; tek adres
```

DEĞİŞKENLER

AX = 1234h

DS = 2000h

BX = ?

döngü = 1/1 yazma

FA = 20100h

BHE' = 0

A0 = 0

D15-D8 = 12h

D7-D0 = 34h

RD' / WR' = 1/0

[20100h] = 34h

[20101h] = 12h

[20102h] = ??

Kod İzleme: Hizalı ve Hizasız Erişim

AX ← ABCDh.

Adım 5 / 10

```
1 mov ax, 2000h
2 mov ds, ax
3 mov ax, 1234h
4 mov [0100h], ax ; çift adres
▶ mov ax, 0ABCDh
6 mov [0101h], ax ; tek adres
7 mov al, [0100h] ; bayt
8 mov bx, [0101h] ; tek adres
```

DEĞİŞKENLER

AX = ABCDh

DS = 2000h

BX = ?

döngü = -

FA = -

BHE' = -

A0 = -

D15-D8 = -

D7-D0 = -

RD' / WR' = 1/1

[20100h] = 34h

[20101h] = 12h

[20102h] = ??

Kod İzleme: Hizalı ve Hizasız Erişim

FA = 20101h **tek**. 1. döngü: BHE'=0, A0=1 → yalnız yüksek banka. AL'deki CDh **D8–D15** üzerinden 20101h'ye yazılır.

Adım 6 / 10

```
1 mov ax, 2000h
2 mov ds, ax
3 mov ax, 1234h
4 mov [0100h], ax ; çift adres
5 mov ax, 0ABCDh
▶ mov [0101h], ax ; tek adres
7 mov al, [0100h] ; bayt
8 mov bx, [0101h] ; tek adres
```

DEĞİŞKENLER

AX = ABCDh DS = 2000h BX = ? döngü = 1/2 yazma

FA = 20101h BHE' = 0 A0 = 1 D15–D8 = CDh

D7–D0 = – RD' / WR' = 1/0 [20100h] = 34h

[20101h] = CDh [20102h] = ??

Kod İzleme: Hizalı ve Hizasız Erişim

2. döngü: FA = 20102h, BHE'=1, A0=0 → yalnız düşük banka. AH'deki ABh **D0-D7** üzerinden yazılır.
Hizasız sözcük = **2 döngü**.

Adım 7 / 10

```
1 mov ax, 2000h
2 mov ds, ax
3 mov ax, 1234h
4 mov [0100h], ax ; çift adres
5 mov ax, 0ABCDh
▶ mov [0101h], ax ; tek adres
7 mov al, [0100h] ; bayt
8 mov bx, [0101h] ; tek adres
```

DEĞİŞKENLER

AX = ABCDh

DS = 2000h

BX = ?

döngü = 2/2 yazma

FA = 20102h

BHE' = 1

A0 = 0

D15-D8 = -

D7-D0 = ABh

RD' / WR' = 1/0

[20100h] = 34h

[20101h] = CDh

[20102h] = ABh

Kod İzleme: Hizalı ve Hizasız Erişim

Bayt okuma, çift adres: BHE'=1, A0=0 → D0–D7. AL ← 34h, AH değişmez → AX = AB34h.

Adım 8 / 10

```
1 mov ax, 2000h
2 mov ds, ax
3 mov ax, 1234h
4 mov [0100h], ax ; çift adres
5 mov ax, 0ABCDh
6 mov [0101h], ax ; tek adres
▶ mov al, [0100h] ; bayt
8 mov bx, [0101h] ; tek adres
```

DEĞİŞKENLER

AX = AB34h

DS = 2000h

BX = ?

döngü = 1/1 okuma

FA = 20100h

BHE' = 1

A0 = 0

D15–D8 = –

D7–D0 = 34h

RD' / WR' = 0/1

[20100h] = 34h

[20101h] = CDh

[20102h] = ABh

Kod İzleme: Hizalı ve Hizasız Erişim

Hizasız sözcük okuma. 1. döngü: 20101h'deki CDh **D8–D15**'ten gelir; işlemci onu BL'ye yerleştirir.

Adım 9 / 10

```
1 mov ax, 2000h
2 mov ds, ax
3 mov ax, 1234h
4 mov [0100h], ax ; çift adres
5 mov ax, 0ABCDh
6 mov [0101h], ax ; tek adres
7 mov al, [0100h] ; bayt
▶ mov bx, [0101h] ; tek adres
```

DEĞİŞKENLER

AX = AB34h

DS = 2000h

BX = ??CDh

döngü = 1/2 okuma

FA = 20101h

BHE' = 0

A0 = 1

D15–D8 = CDh

D7–D0 = –

RD' / WR' = 0/1

[20100h] = 34h

[20101h] = CDh

[20102h] = ABh

Kod İzleme: Hizalı ve Hizasız Erişim

2. döngü: 20102h'deki ABh **D0–D7**'den gelir → BH. Sonuç BX = **ABCDh** (little-endian).

Adım 10 / 10

```
1 mov ax, 2000h
2 mov ds, ax
3 mov ax, 1234h
4 mov [0100h], ax ; çift adres
5 mov ax, 0ABCDh
6 mov [0101h], ax ; tek adres
7 mov al, [0100h] ; bayt
▶ mov bx, [0101h] ; tek adres
```

DEĞİŞKENLER

AX = **AB34h**

DS = **2000h**

BX = **ABCDh**

döngü = **2/2 okuma**

FA = **20102h**

BHE' = **1**

A0 = **0**

D15–D8 = **–**

D7–D0 = **ABh**

RD' / WR' = **0/1**

[20100h] = **34h**

[20101h] = **CDh**

[20102h] = **ABh**

Yol Döngüsü ve T Durumları

- İşlemcinin bellek veya G/Ç ile her veri alışverişi bir **yol döngüsüdür** (bus cycle).
- Bir yol döngüsü en az **dört saat turu** sürer: **T1, T2, T3, T4** (T durumları).
- Pinlerin işlevi T durumuna göre değişir: T1'de adres, T2–T4'te veri/durum.
- Yavaş bellek için T3 ile T4 arasına **bekleme durumları (Tw)** eklenir.
- İşlemci yolu kullanmadığı turlar **boşta** durumlarıdır (Ti).

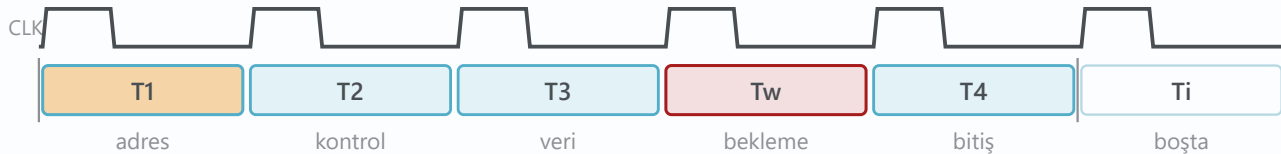
5 MHz örneği

$$T = 1 / 5 \text{ MHz} = 200 \text{ ns}$$

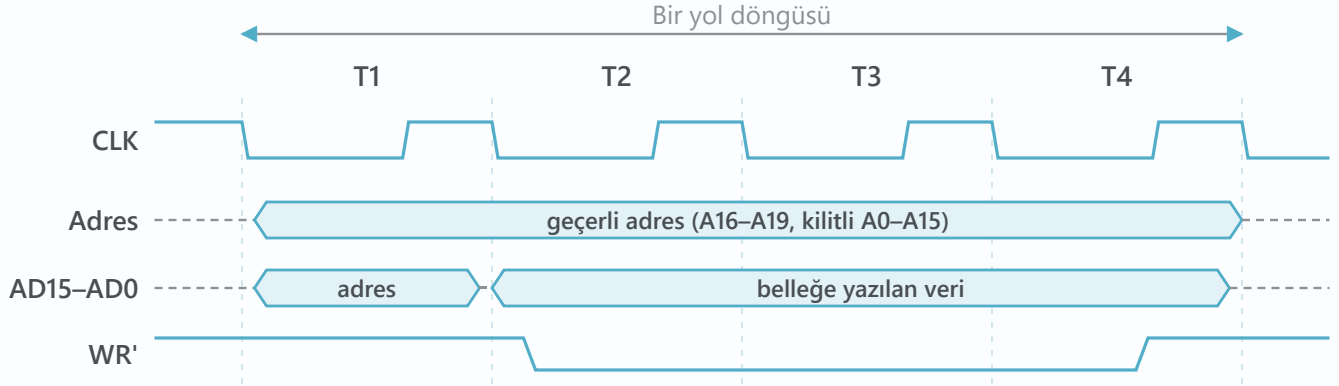
$$\text{Yol döngüsü} = 4 \times 200 = 800 \text{ ns}$$

En fazla $1 / 800 \text{ ns} = 1,25 \text{ milyon}$ yol döngüsü/s

Sözcük başına 2 bayt $\rightarrow \approx 2,5 \text{ MB/s}$ yol bant genişliği



Yazma Yol Döngüsü

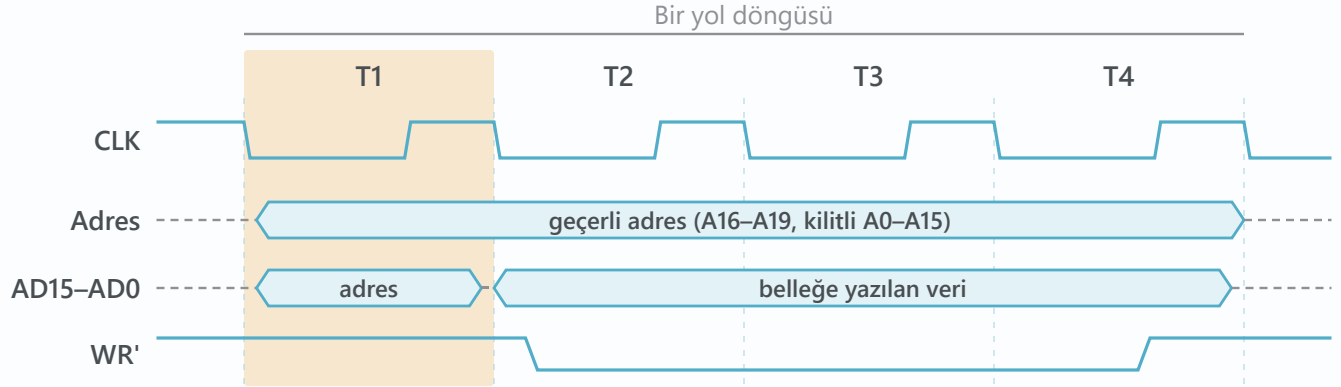


Yazma döngüsü dört T durumundan oluşur. Saat düşen kenarı her T durumunun başıdır.

Adım 1 / 5

Yazmada adres T1'de, veri T2-T4 boyunca aynı pinlerde: arada bekleme yoktur, çünkü verinin kaynağı işlemcinin kendisidir.

Yazma Yol Döngüsü

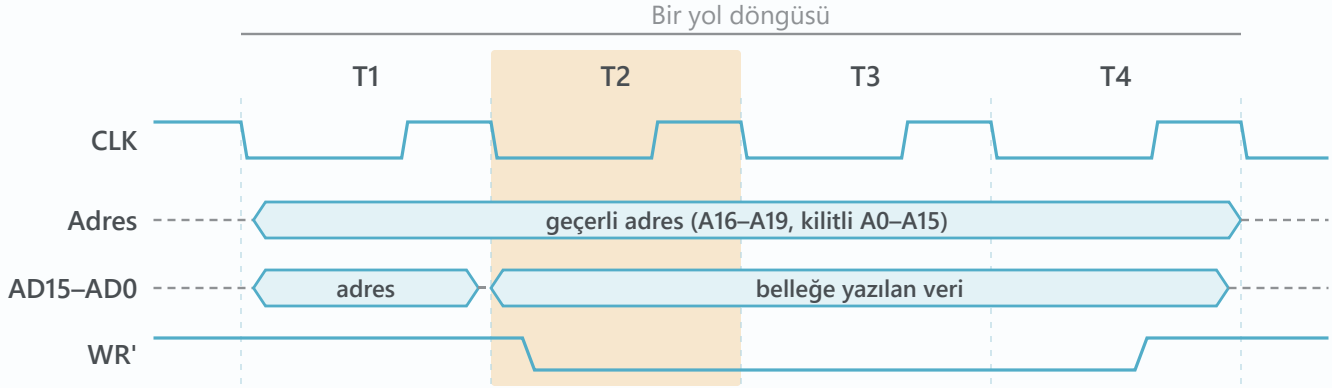


T1: yazılacak hedef adres AD ve A pinlerine konur; ALE ile dış kilitlere alınır.

Adım 2 / 5

Yazmada adres T1'de, veri T2–T4 boyunca aynı pinlerde: arada bekleme yoktur, çünkü verinin kaynağı işlemcinin kendisidir.

Yazma Yol Döngüsü

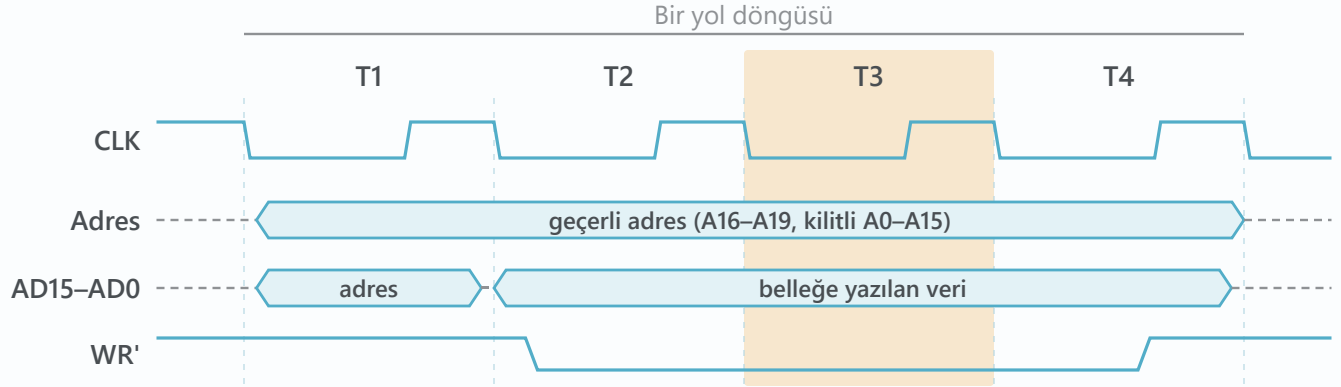


T2: AD pinleri artık **yazılacak veriyi** taşır; $WR' = 0$ olur.

Adım 3 / 5

Yazmada adres T1'de, veri T2-T4 boyunca aynı pinlerde: arada bekleme yoktur, çünkü verinin kaynağı işlemcinin kendisidir.

Yazma Yol Döngüsü



T3: veri ve WR' kararlı kalır; bellek veriyi yakalar (READY = 0 ise burada Tw eklenir).

Adım 4 / 5

Yazmada adres T1'de, veri T2-T4 boyunca aynı pinlerde: arada bekleme yoktur, çünkü verinin kaynağı işlemcinin kendisidir.

Yazma Yol Döngüsü

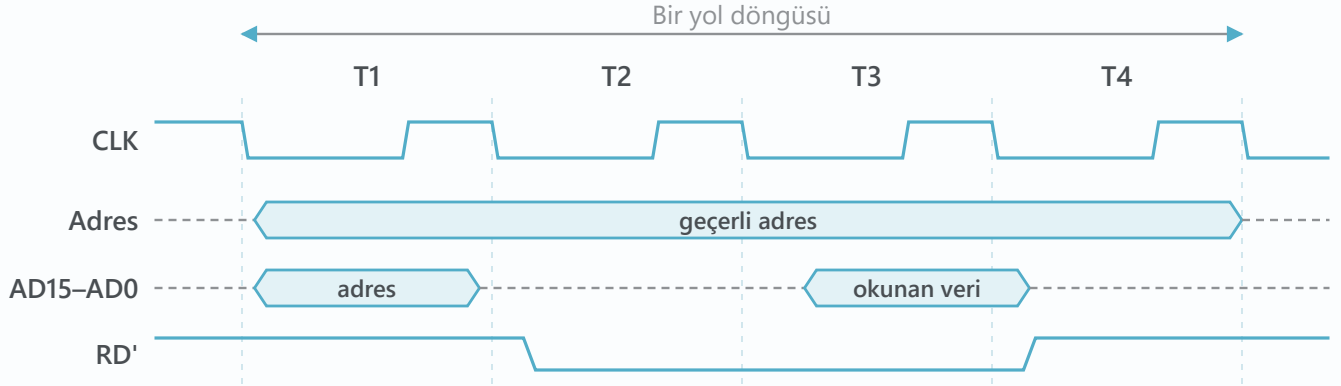


T4: $WR' = 1$ 'e döner, yükselen kenarında bellek yazmayı tamamlar; yol bir sonraki döngüye hazırlanır.

Adım 5 / 5

Yazmada adres T1'de, veri T2-T4 boyunca aynı pinlerde: arada bekleme yoktur, çünkü verinin kaynağı işlemcinin kendisidir.

Okuma Yol Döngüsü

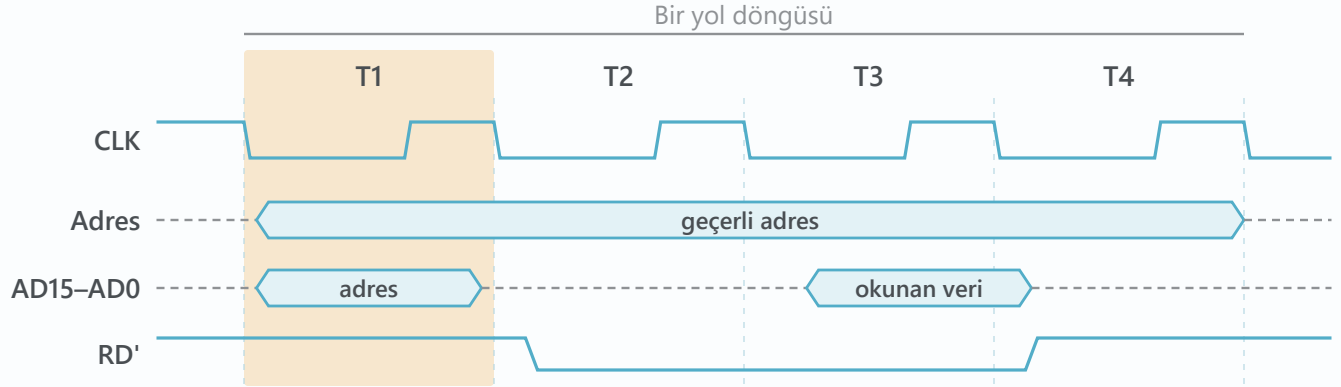


Okuma döngüsü de dört T durumudur, ama AD pinlerinde adres ile veri arasında bir **boşluk** (float) vardır.

Adım 1 / 5

T2'deki yüzer durum "yol dönüşü" (bus turnaround) içindir: işlemci adresi sürmeyi bırakmadan bellek veriyi sürmeye başlarsa iki çıkış çakışır.

Okuma Yol Döngüsü

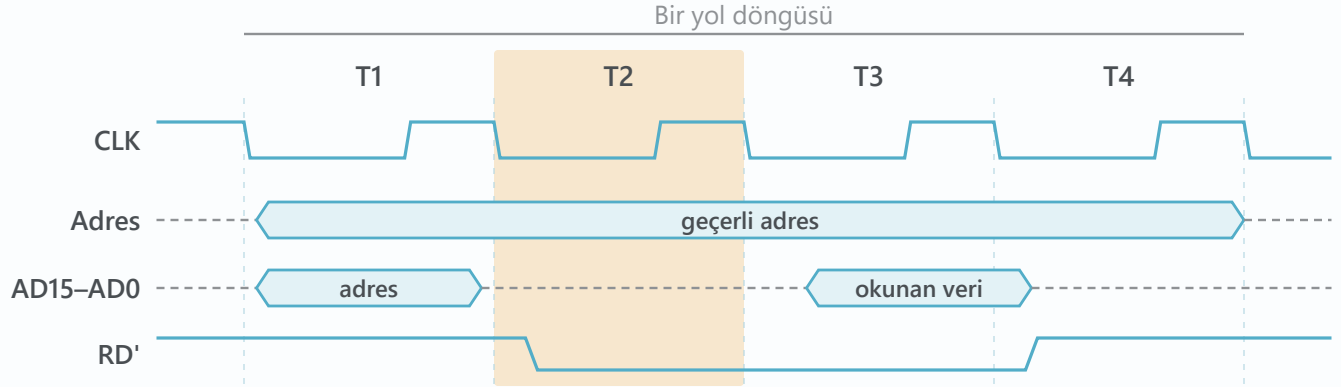


T1: yazmadaki gibi okunacak adres pinlere konur ve kilitlenir.

Adım 2 / 5

T2'deki yüzer durum "yol dönüşü" (bus turnaround) içindir: işlemci adresi sürmeyi bırakmadan bellek veriyi sürmeye başlarsa iki çıkış çakışır.

Okuma Yol Döngüsü

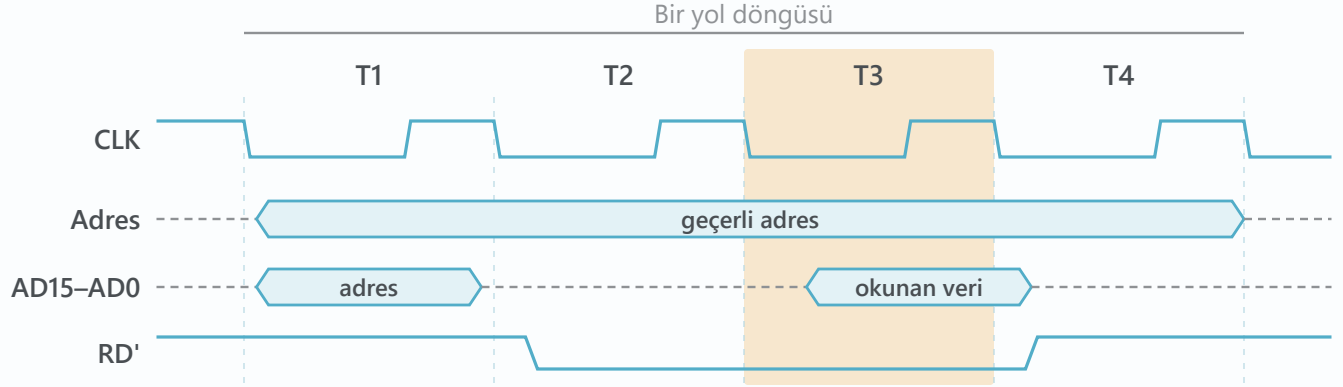


T2: 8086 AD çıkış sürücülerini **üç durumlu** (tri-state, yüksek empedans) yapar → bellek yolu sürebilir. RD' = 0.

Adım 3 / 5

T2'deki yüzer durum "yol dönüşü" (bus turnaround) içindir: işlemci adresi sürmeyi bırakmadan bellek veriyi sürmeye başlarsa iki çıkış çakışır.

Okuma Yol Döngüsü

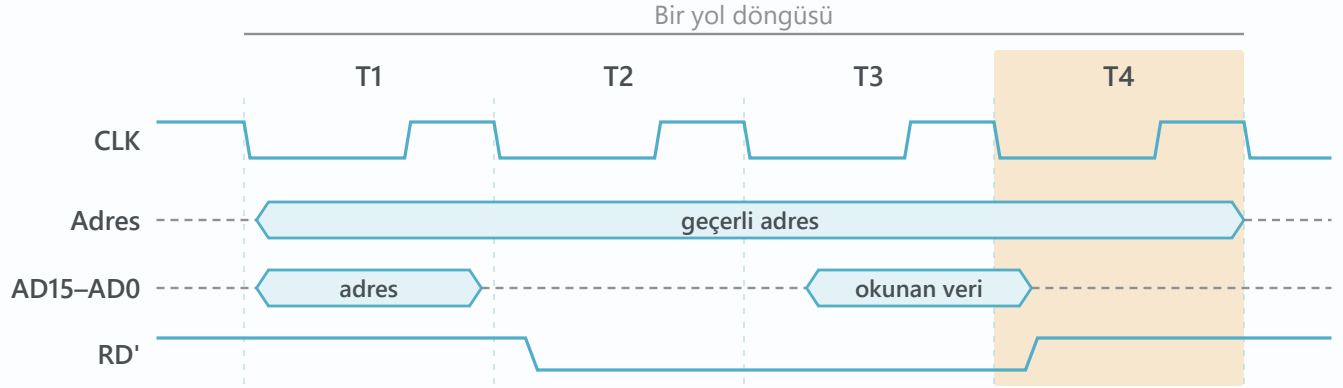


T3: bellek (veya G/Ç aygıtı) istenen veriyi veri yoluna koyar.

Adım 4 / 5

T2'deki yüzer durum "yol dönüşü" (bus turnaround) içindir: işlemci adresi sürmeyi bırakmadan bellek veriyi sürmeye başlarsa iki çıkış çakışır.

Okuma Yol Döngüsü

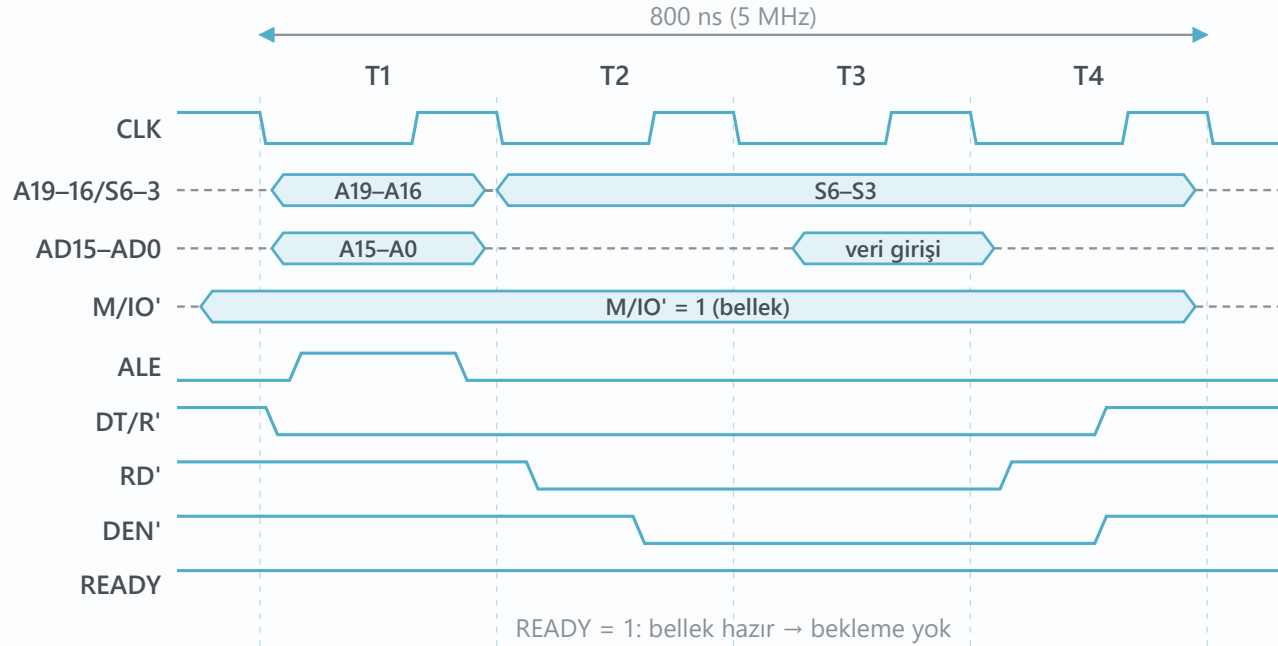


T4: T4'ün başında 8086 veriyi pinlerden okur (örnekler); RD' = 1, bellek yolu bırakır.

Adım 5 / 5

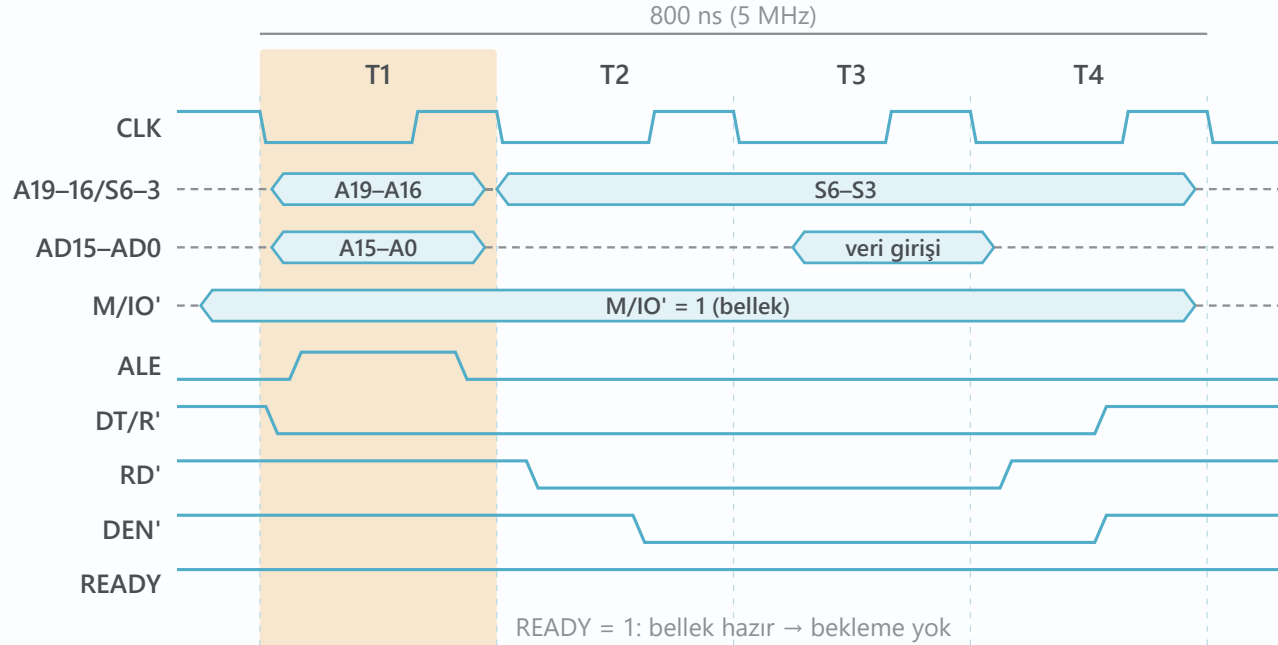
T2'deki yüzer durum "yol dönüşü" (bus turnaround) içindir: işlemci adresi sürmeyi bırakmadan bellek veriyi sürmeye başlarsa iki çıkış çakışır.

Okuma İçin Yol Zamanlaması



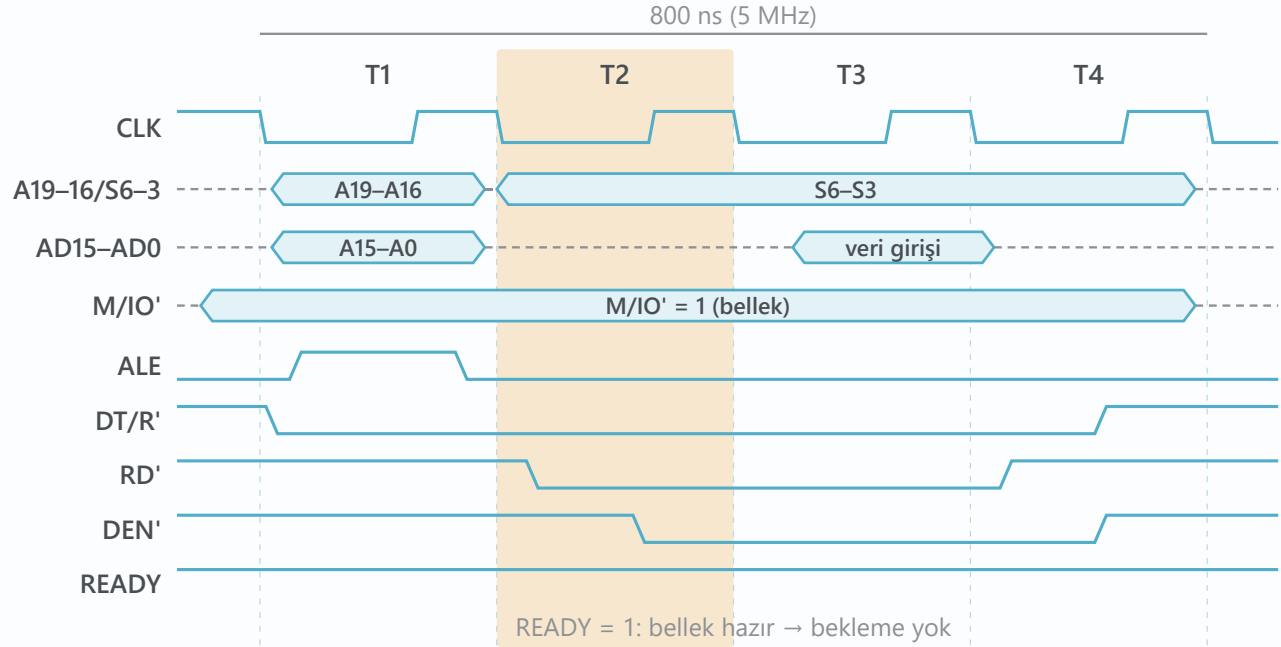
Minimum modda bir bellek okuma döngüsünde tüm kontrol sinyalleri.

Okuma İçin Yol Zamanlaması



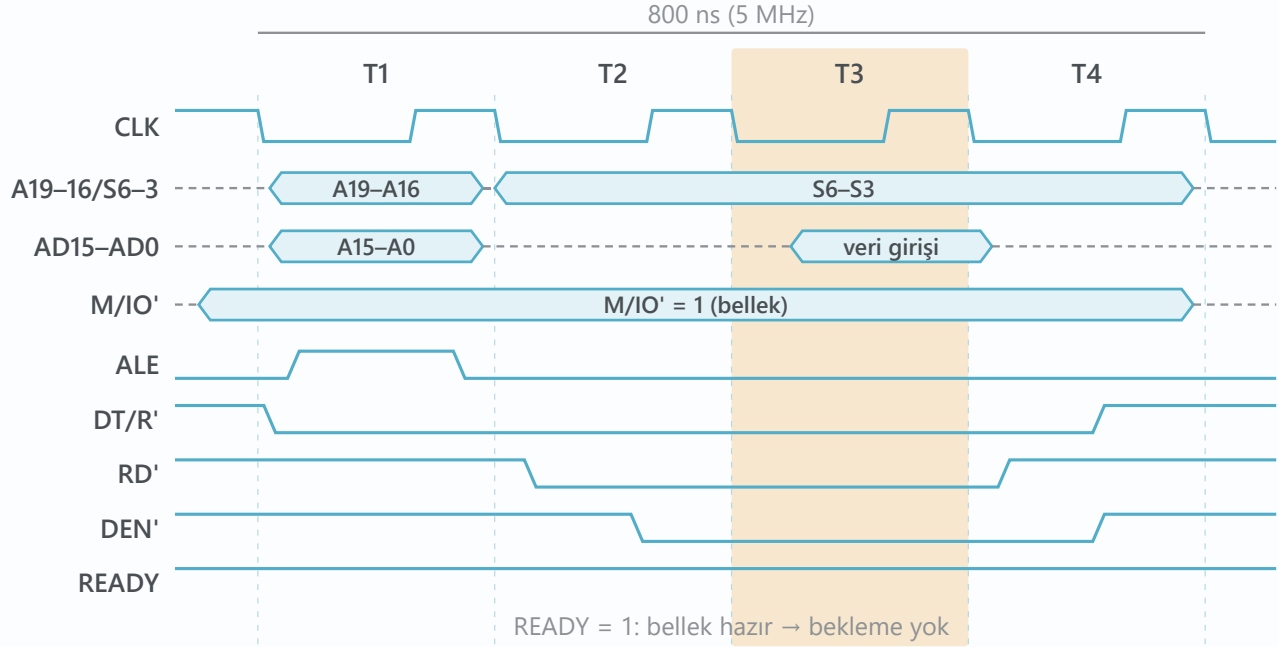
T1: M/IO' = 1 (bellek), adres pinlerde; **ALE** darbesi adresi kilitler; DT/R' = 0 (alma yönü).

Okuma İçin Yol Zamanlaması



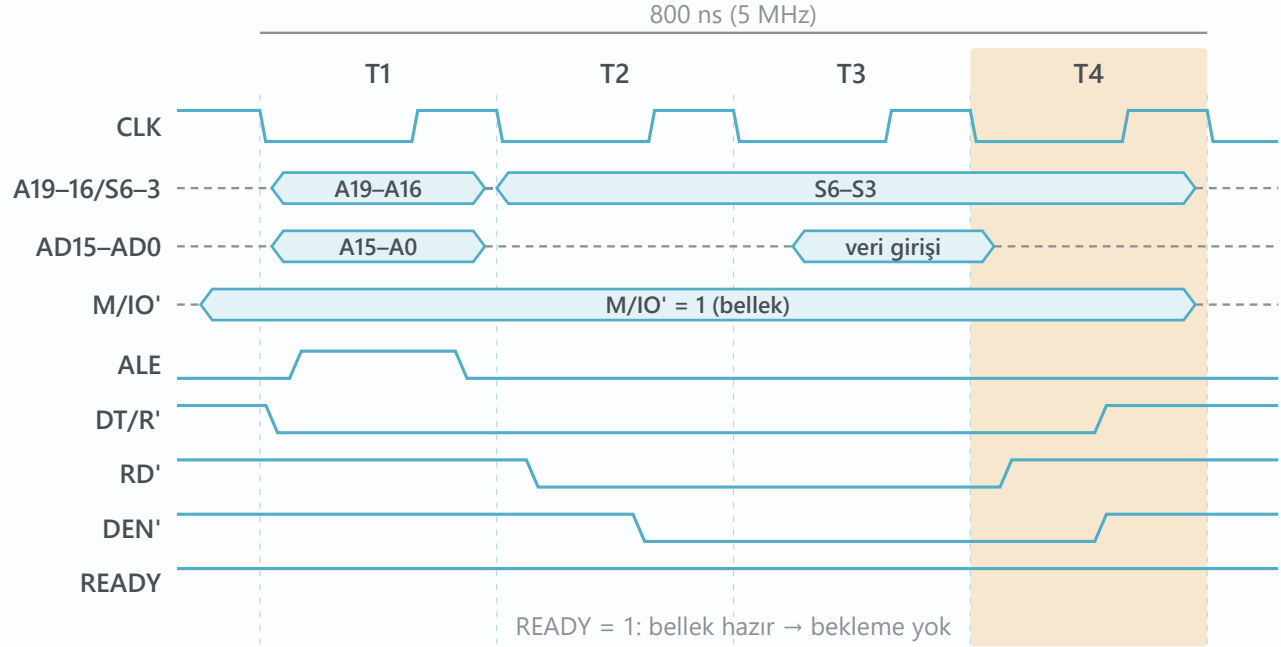
T2: AD yüzer, A19-A16 yerine S6-S3; **RD'** = 0 ve ardından **DEN'** = 0 → alıcı-verici açılır.

Okuma İçin Yol Zamanlaması



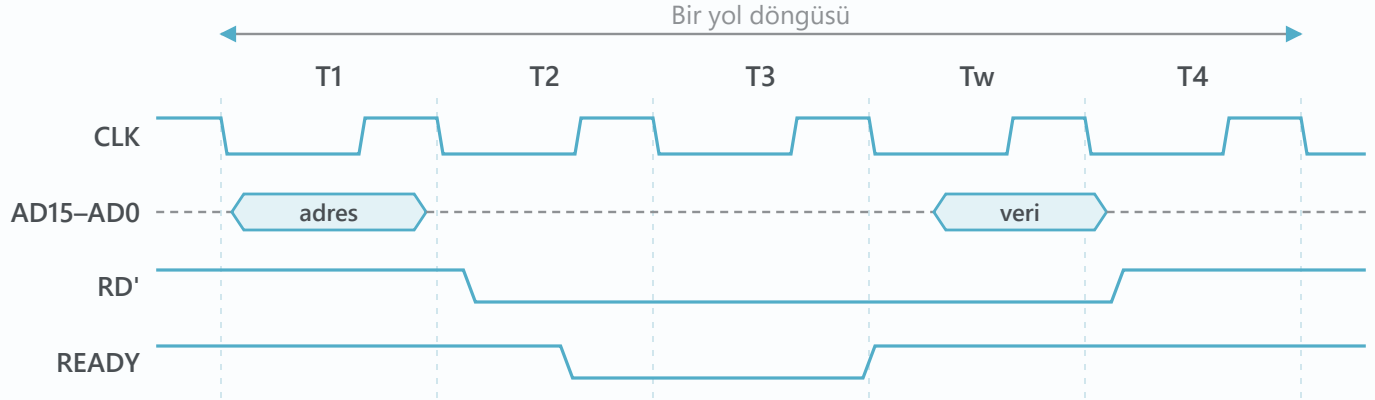
T3: bellek veriyi koyar. İşlemci T3'ün başında **READY**'yi örnekler: 1 ise T4'e geçer.

Okuma İçin Yol Zamanlaması



T4: veri örneklenir; RD', DEN', DT/R' pasife döner; bir sonraki döngü için M/IO' değişebilir.

READY ve Bekleme Durumu (Tw)

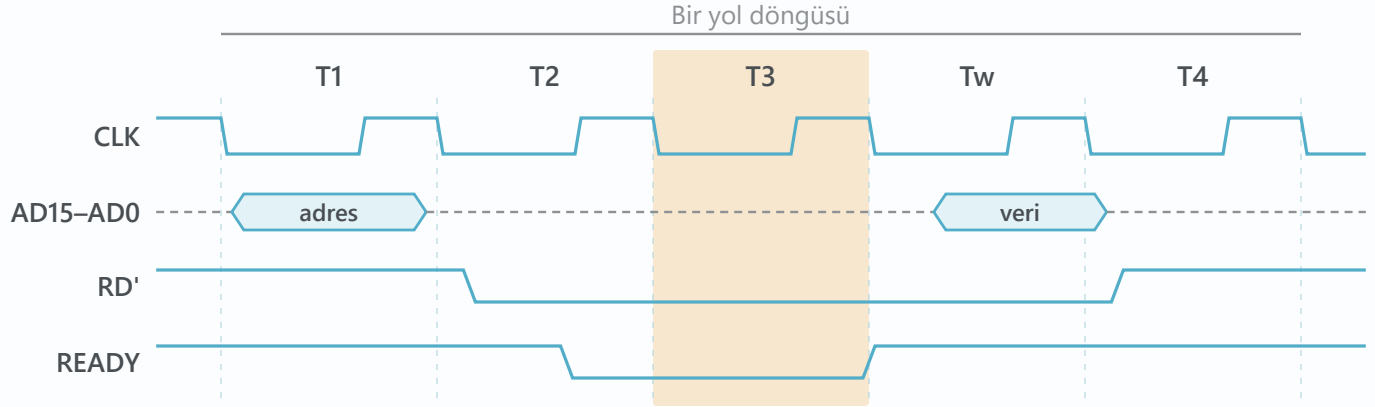


Yavaş bir bellek, verisini T3'e kadar hazırlayamıyorsa **READY** sinyalini 0 yapar.

Adım 1 / 4

READY, 8284A'nın RDY girişinden saate senkronlanarak 8086'ya verilir (aktif yüksek).

READY ve Bekleme Durumu (T_w)

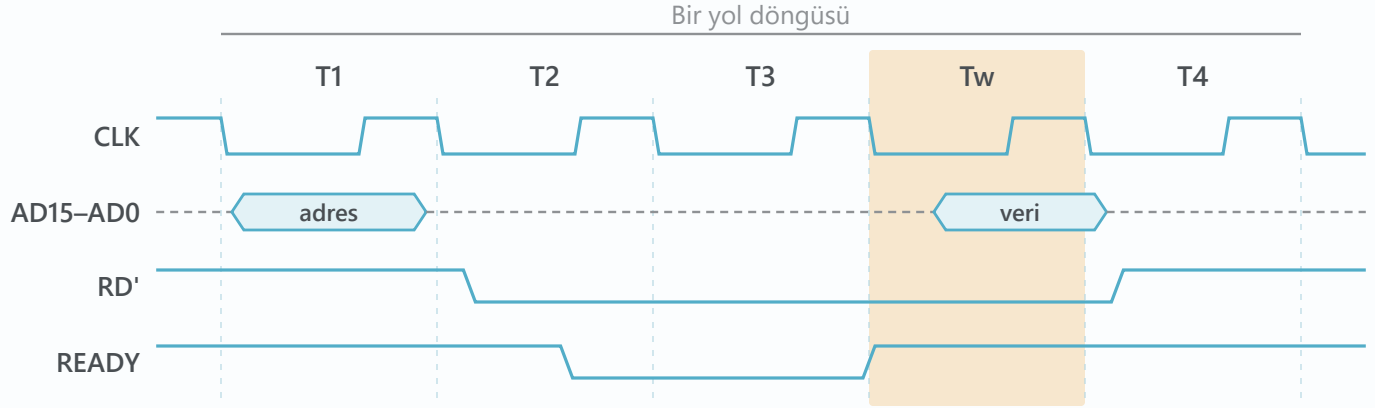


T3'te $READY = 0$ görülür → işlemci T4'e geçmez, bir **T_w** (bekleme) durumu ekler. Tüm sinyaller olduğu gibi kalır.

Adım 2 / 4

READY, 8284A'nın RDY girişinden saate senkronlanarak 8086'ya verilir (aktif yüksek).

READY ve Bekleme Durumu (Tw)

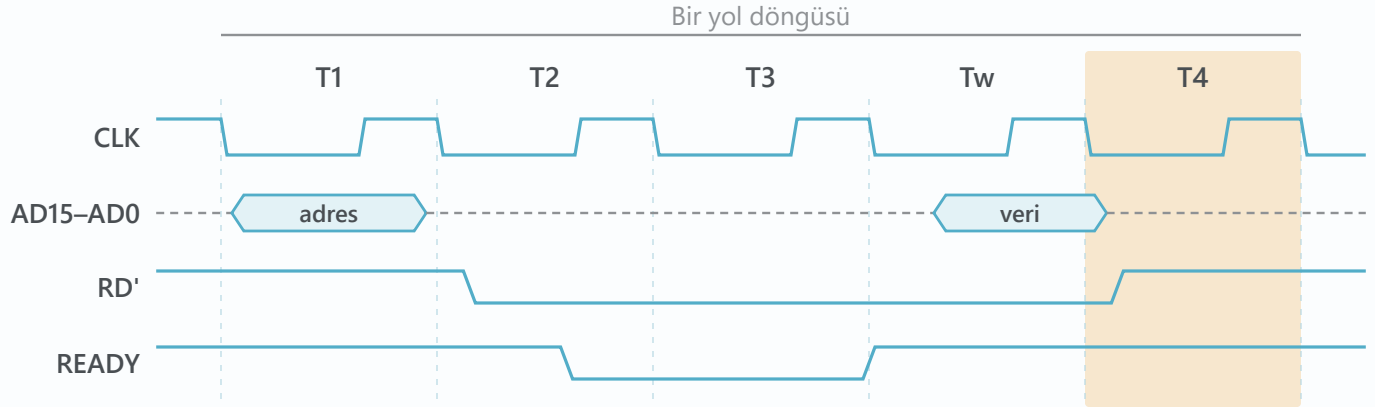


Tw'nin sonunda READY tekrar örneklenir; bu kez 1 → bellek hazır. Gerekirse birden fazla Tw eklenebilir.

Adım 3 / 4

READY, 8284A'nın RDY girişinden saate senkronlanarak 8086'ya verilir (aktif yüksek).

READY ve Bekleme Durumu (Tw)



T4: veri okunur. Döngü $5 \times 200 = 1000$ ns sürdü (bir bekleme durumu = +200 ns).

Adım 4 / 4

READY, 8284A'nın RDY girişinden saate senkronlanarak 8086'ya verilir (aktif yüksek).

Kaç Bekleme Durumu Gerekir?

- Okumada adres T1'in başından en geç **T_{CLAV}** sonra geçerli olur; veri T3'ün sonunda, T4'e geçen saat kenarında örneklenir ve bundan **T_{DVCL}** önce hazır olmalıdır.
- Arada adres kilidi, adres çözücü ve veri alıcı-vericisinin gecikmeleri (**t_{tampon}**) de harcanır.
- Her Tw pencereyi bir T uzatır.

$$t_{izin} = 3 \cdot T - T_{CLAV} - T_{DVCL} - t_{tampon}$$

$$\text{Koşul: } t_{erişim} \leq t_{izin} + n \cdot T \rightarrow n = \lceil (t_{erişim} - t_{izin}) / T \rceil \text{ (negatifse } n = 0)$$

RD' T2'de düştüğü için RD' → veri süresine (çıkış etkinleştirme) ikinci bir koşul daha vardır; burada yalnız adres → veri koşulu hesaplanıyor.

Örnek: 5 MHz 8086

T = 200 ns; veri sayfası: $T_{CLAV} \leq 110$ ns,
 $T_{DVCL} \geq 30$ ns; $t_{tampon} = 40$ ns kabul edilsin.

$$t_{izin} = 600 - 110 - 30 - 40 = \mathbf{420 \text{ ns}}$$

Bellek $t_{erişim} = 700$ ns:

$$n = \lceil (700 - 420) / 200 \rceil = \lceil 1,4 \rceil = \mathbf{2 \text{ Tw}}$$

Sinama: 1 Tw → 620 ns < 700 X; 2 Tw → 820 ns ≥ 700 ✓

$$\text{Döngü} = (4 + 2) \times 200 = \mathbf{1200 \text{ ns}}$$

$t_{erişim} = 400$ ns olsaydı: $400 \leq 420 \rightarrow \mathbf{0 \text{ Tw}}$.

DURUM PİNLERİ

S2', S1', S0' Durum Pinleri

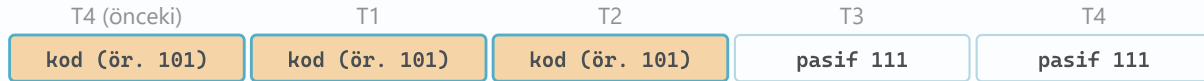
- Maksimum modda (pin 26–28) yol döngüsünün **türünü** kodlar.
- T4 (önceki döngü), T1 ve T2 sırasında aktiftir.
- T3'te veya READY = 1 olan Tw'de **pasif** duruma (1, 1, 1) döner.
- **8288 yol denetleyicisi** bunları çözerek bellek ve G/Ç erişim kontrol sinyallerini (MRDC', MWTC', IORC' ...) üretir.
- T4'te pasif durumdan herhangi bir koda geçiş, yeni bir **yol döngüsünün başladığını** gösterir.

Seviye gösterimi

Düşük (low) sinyal = **0** $\approx$ 0 V

Yüksek (high) sinyal = **1** $\approx$ +5 V

Durum pinleri aktif düşüktür; "111 = pasif" bu yüzden "hiçbir şey yapılmıyor" anlamına gelir.



8288 kodu T4–T2 arasında çözer; T4'te 111'den çıkış yeni döngüyü haber verir

DURUM PİNLERİ

S2' S1' S0' Kodları

S2'	S1'	S0'	Döngü	8288 komutu
0	0	0	Kesme kabul (interrupt acknowledge)	INTA'
0	0	1	G/Ç portu oku	IORC'
0	1	0	G/Ç portuna yaz	IOWC', AIOWC'
0	1	1	Durdurma (HLT)	yok
1	0	0	Kod erişimi (komut getirme)	MRDC'
1	0	1	Bellek oku	MRDC'
1	1	0	Belleğe yaz	MWTC', AMWC'
1	1	1	Pasif	yok

Okuma ipucu

S2' = 1 → **bellek**, S2' = 0 → G/Ç / özel.

S1' ve S0': 01 → okuma, 10 → yazma.

Bu yüzden S2' minimum moddaki M/IO' ile, S1' DT/R' ile aynı pini paylaşır.

DURUM PİNLERİ

S3–S6 ve BHE'/S7 Durum Bitleri

A16/S3, A17/S4, A18/S5, A19/S6: T1'den sonra adres yerine durum bitleri.

S4 (A17)	S3 (A16)	Kullanılan kesim
0	0	ES (ek kesim)
0	1	SS (yığın)
1	0	CS veya kesim yok (G/Ç, kesme vektörü)
1	1	DS (veri)

Diğer durum bitleri

- **S5** (A18): IF kesme bayrağının o anki değeri; her saat turunun başında güncellenir.
- **S6** (A19): her zaman 0 (8086 yolu kullanıyor).
- **S7** (BHE'): kullanılmayan yedek durum biti.

S3–S4, sistemin her kesim için ayrı 1 MB bellek bağlamasına olanak verir (ör. yığın ayrı bankada).

Kod İzleme: Komutlar ve Durum Kodları

```
1 org 100h
```

```
► mov bx, 0200h
```

```
3 mov ax, [bx]      ; bellek oku
4 mov [bx+2], ax    ; belleğe yaz
5 in al, 60h        ; G/Ç oku
6 out 61h, al       ; G/Ç yaz
7 push ax           ; yığına yaz
8 hlt               ; dur
```

8086 durum pinleri



→ CS ya da yok

8288 yol denetleyicisi

MRDC'

kod getirme

adres: CS:0100h

AX = 0000h · SP = FFEh

Her döngünün başında 8086, S2' S1' S0' pinlerine döngü türünü koyar. `mov bx, 0200h` yalnız **kod getirme** ister: **100**; S4 S3 = 10 → CS.

Adım 1 / 7

DEĞİŞKENLER

S2' S1' S0' = 100

S4 S3 = 10

komut = MRDC'

AX = 0000h

Komut baytlarını getiren 100 (kod getirme) döngüleri ayrıca yapılır.

Kod İzleme: Komutlar ve Durum Kodları

```
1 org 100h
2 mov bx, 0200h
3 ► mov ax, [bx] ; bellek oku
4 mov [bx+2], ax ; belleğe yaz
5 in al, 60h ; G/Ç oku
6 out 61h, al ; G/Ç yaz
7 push ax ; yığına yaz
8 hlt ; dur
```

8086 durum pinleri

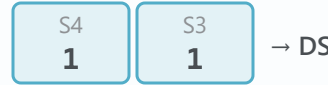


8288 yol denetleyicisi
MRDC'

bellek oku

adres: DS:0200h

AX = 1234h · SP = FFEh



`mov ax, [bx]` : bellek okuma **101**. 8288 **MRDC'** üretir. Veri kesimi kullanıldığı için S4 S3 = **11** (DS). AX = 1234h.

Adım 2 / 7

DEĞİŞKENLER

S2'S1'S0' = **101**

S4S3 = **11**

komut = **MRDC'**

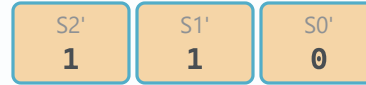
AX = **1234h**

Komut baytlarını getiren 100 (kod getirme) döngüleri ayrıca yapılır.

Kod İzleme: Komutlar ve Durum Kodları

```
1 org 100h
2 mov bx, 0200h
3 mov ax, [bx]      ; bellek oku
4 ► mov [bx+2], ax  ; belleğe yaz
5 in al, 60h        ; G/Ç oku
6 out 61h, al       ; G/Ç yaz
7 push ax           ; yığına yaz
8 hlt               ; dur
```

8086 durum pinleri



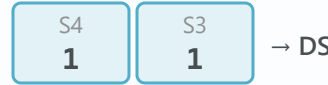
8288 yol denetleyicisi

MWTC', AMWC'

belleğe yaz

adres: DS:0202h

AX = 1234h · SP = FFEh



`mov [bx+2], ax` : belleğe yazma **110** → 8288 **MWTC'** (ve bir saat önce AMWC'). Kesim yine DS.

Adım 3 / 7

DEĞİŞKENLER

S2' S1' S0' = **110**

S4 S3 = **11**

komut = **MWTC'**

AX = **1234h**

Komut baytlarını getiren 100 (kod getirme) döngüleri ayrıca yapılır.

Kod İzleme: Komutlar ve Durum Kodları

```
1 org 100h
2 mov bx, 0200h
3 mov ax, [bx]      ; bellek oku
4 mov [bx+2], ax    ; belleğe yaz
5 ► in al, 60h      ; G/Ç oku
6 out 61h, al       ; G/Ç yaz
7 push ax           ; yığına yaz
8 hlt               ; dur
```

8086 durum pinleri



8288 yol denetleyicisi

IORC'

G/Ç oku

adres: port 60h

AX = 125Ah · SP = FFEh



→ CS ya da yok

in al, 60h : G/Ç okuma **001** → **IORC'**. G/Ç'de kesim yok:
S4 S3 = 10. AL = 5Ah.

Adım 4 / 7

DEĞİŞKENLER

S2'S1'S0' = 001

S4S3 = 10

komut = IORC'

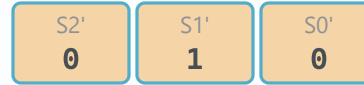
AX = 125Ah

Komut baytlarını getiren 100 (kod getirme) döngüleri ayrıca yapılır.

Kod İzleme: Komutlar ve Durum Kodları

```
1 org 100h
2 mov bx, 0200h
3 mov ax, [bx]      ; bellek oku
4 mov [bx+2], ax    ; belleğe yaz
5 in al, 60h        ; G/Ç oku
6 out 61h, al       ; G/Ç yaz
7 push ax           ; yığına yaz
8 hlt               ; dur
```

8086 durum pinleri



8288 yol denetleyicisi

IOWC', AIOWC'

G/Ç yaz

adres: port 61h

AX = 125Ah · SP = FFEh



→ CS ya da yok

out 61h, al : G/Ç yazma **010** → IOWC'. S2' = 0
olduğundan bellek bu döngüye karışmaz.

Adım 5 / 7

DEĞİŞKENLER

S2'S1'S0' = **010**

S4S3 = **10**

komut = IOWC'

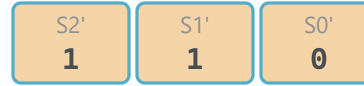
AX = **125Ah**

Komut baytlarını getiren 100 (kod getirme) döngüleri ayrıca yapılır.

Kod İzleme: Komutlar ve Durum Kodları

```
1 org 100h
2 mov bx, 0200h
3 mov ax, [bx]      ; bellek oku
4 mov [bx+2], ax    ; belleğe yaz
5 in al, 60h        ; G/Ç oku
6 out 61h, al       ; G/Ç yaz
▶ push ax           ; yığına yaz
8 hlt               ; dur
```

8086 durum pinleri



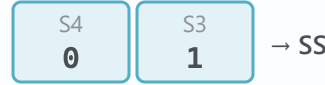
8288 yol denetleyicisi

MWTC', AMWC'

belleğe yaz

adres: SS:FFFCh

AX = 125Ah · SP = FFFCh



push ax da bir bellek yazmasıdır (**110**), ama yığın kesimi kullanılır: S4 S3 = **01** (SS). Sistem yığını ayrı bir bankaya koyabilir.

Adım 6 / 7

DEĞİŞKENLER

S2'S1'S0' = **110**

S4S3 = **01**

komut = **MWTC'**

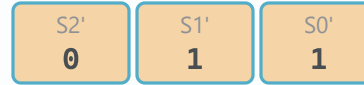
AX = **125Ah**

Komut baytlarını getiren 100 (kod getirme) döngüleri ayrıca yapılır.

Kod İzleme: Komutlar ve Durum Kodları

```
1 org 100h
2 mov bx, 0200h
3 mov ax, [bx]      ; bellek oku
4 mov [bx+2], ax    ; belleğe yaz
5 in al, 60h        ; G/Ç oku
6 out 61h, al       ; G/Ç yaz
7 push ax           ; yığına yaz
▶ hlt               ; dur
```

8086 durum pinleri



8288 yol denetleyicisi
yok

durdurma (HLT)

adres: -

→ CS ya da yok

AX = 125Ah · SP = FFFCh

hlt : **011** durdurma kodu; 8288 komut üretmez. Döngü aralarında pinler **111** (pasif) değerine döner.

Adım 7 / 7

DEĞİŞKENLER

S2'S1'S0' = **011**

S4S3 = **10**

komut = **yok**

AX = **125Ah**

Komut baytlarını getiren 100 (kod getirme) döngüleri ayrıca yapılır.

DURUM PİNLERİ

Demo: Durum Kodu Çözücü

Pinlere tıklayarak 0/1 yap. 8288'in ve sistemin bu kodu nasıl yorumladığını gör.

S2' = 1

S1' = 0

S0' = 1

S4 = 1

S3 = 1

S2' S1' S0'	101 → Bellek oku
8288 komutu	MRDC'
Açıklama	Ör. MOV AL, [BX].
S4 S3	11 → DS (veri)

S2' = 1 → bellek döngüsü

DURUM PİNLERİ

QS1, QS0: Kuyruk Durumu

- Maksimum modda (pin 24–25) önden getirme **kuyruğunun durumunu** dışarı bildirir.
- Kuyruk işlemini izleyen saat turunda geçerlidir.
- 8087 yardımcı işlemcisi bu pinlerle 8086'nın hangi komutu yürüttüğünü izler ve kendi komutlarını (ESC) yakalar.

QS1	QS0	Durum
0	0	İşlem yok (kuyruk değişmedi)
0	1	Kuyruktan işlem kodunun ilk baytı alındı
1	0	Kuyruk boşaltıldı (dallanma)
1	1	Kuyruktan sonraki bayt alındı

KONTROL PİNLERİ

RD', WR' ve M/IO'

RD' · Okuma (pin 32)

- Çıkış, **aktif düşük**.
- İşlemcinin bellekten veya G/Ç'den okuma yaptığını bildirir.
- Okuma döngüsünün T2, T3 ve Tw'lerinde 0.

WR' · Yazma (pin 29, min. mod)

- Çıkış, aktif düşük; yazma döngüsünün T2, T3, Tw'lerinde 0.

M/IO' · Bellek / G-Ç (pin 28, min. mod)

- Bellek ve G/Ç erişimlerini ayırır.
- **1** → bellek işlemi (M aktif), **0** → G/Ç işlemi (IO' aktif).
- Önceki döngünün T4'ünde geçerli olur, döngü boyunca kalır.
- 8088'de ters kutupludur: IO/M'.

M/IO'	RD'	WR'	İşlem	Örnek komut
1	0	1	bellek okuma	MOV AL, [BX]
1	1	0	bellek yazma	MOV [BX], AL
0	0	1	G/Ç okuma	IN AL, 60h
0	1	0	G/Ç yazma	OUT 43h, AL

KONTROL PİNLERİ

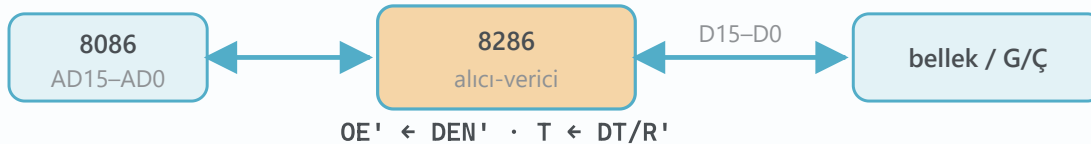
DT/R' ve DEN'

DT/R' · Veri gönder/al (pin 27)

- 8286/8287 veri yolu **alıcı-vericisi** (transceiver) kullanan sistemlerde gereklidir.
- Alıcı-verici: verici ve alıcının tek pakette birleşimi.
- 1** → veri işlemciden dışarı (yazma), **0** → dışarıdan işlemciye (okuma).
- Veri akışının yönünü alıcı-vericinin T girişi belirler.

DEN' · Veri etkinleştirme (pin 26)

- Alıcı-vericiye **çıkış etkinleştirme** (OE') olarak bağlanır.
- Bellek ve G/Ç erişimlerinde ve INTA döngülerinde **aktif düşük**.
- T1'de 1 kalır: AD pinlerindeki adres veri yoluna sızmasın.



DEN' = 0 iken alıcı-verici açık; yönü DT/R' seçer. Okumada (DT/R' = 0) bellek → CPU, yazmada (DT/R' = 1) CPU → bellek.

Doğru gösterim DT/R'dir: R aktif düşüktür.

Komuttan Yol Döngülerine: OUT ve IN

Döngü 1, T1: komut baytları bellekten okunur. $FA = 01A70h + 0018h = 01A88h$ (çift) $\rightarrow BHE' = 0, A0 = 0$: **tek döngüde bir sözcük**. ALE darbesi adresi kilitler; okuma olduğu için $DT/R' = 0$.

Adım 1 / 8

```
1 ; CS = 01A7h, AL = 12h
2 ; FA      bayt   komut
▶ 01A88h E6 59 out 59h, al
4 01A8Ah E4 9A in  al, 9Ah
```

DEĞİŞKENLER

döngü = 1: kod getirme

T = T1

A19-A0 = 01A88h

BHE' = 0

A0 = 0

D15-D8 = -

D7-D0 = -

M/IO' = 1

ALE = 1

RD' = 1

WR' = 1

DT/R' = 0

DEN' = 1

AL = 12h

Basitleştirilmiş sıralı model: gerçekte BIU, kuyrukta yer açıldıkça kodu **önden getirir**; döngü 3 döngü 2'den önce de olabilir. Her döngünün T1-T4 ayrıntısı: [Yazma](#) ve [Okuma Yol Döngüsü](#). 8 bitlik birimi yalnız D7-D0'a bağlı olan sistemde tek port adresi kullanılamaz.

Komuttan Yol Döngülerine: OUT ve IN

T2–T4: $RD' = 0$, $DEN' = 0$. Bellek sözcüğü verir: çift adresteki E6h (işlem kodu) **D7–D0**'dan, tek adresteki 59h (port) **D15–D8**'den gelir.

Adım 2 / 8

```
1 ; CS = 01A7h, AL = 12h
2 ; FA      bayt   komut
▶ 01A88h E6 59 out 59h, al
4 01A8Ah E4 9A in  al, 9Ah
```

DEĞİŞKENLER

döngü = 1: kod getirme

T = T2–T4

A19–A0 = 01A88h

BHE' = 0

A0 = 0

D15–D8 = 59h

D7–D0 = E6h

M/IO' = 1

ALE = 0

RD' = 0

WR' = 1

DT/R' = 0

DEN' = 0

AL = 12h

Basitleştirilmiş sıralı model: gerçekte BIU, kuyrukta yer açıldıkça kodu **önden getirir**; döngü 3 döngü 2'den önce de olabilir. Her döngünün T1–T4 ayrıntısı: [Yazma](#) ve [Okuma Yol Döngüsü](#). 8 bitlik birimi yalnız D7–D0'a bağlı olan sistemde tek port adresi kullanılamaz.

Komuttan Yol Döngülerine: OUT ve IN

Döngü 2, T1: M/IO' = 0 → G/Ç. Adres yalnız A7–A0'da anlamlı (59h); A15–A8 ve A19–A16 = 0. Port **tek** adres → BHE' = 0, A0 = 1. Yazma: DT/R' = 1.

Adım 3 / 8

```
1 ; CS = 01A7h, AL = 12h
2 ; FA      bayt   komut
▶ 01A88h E6 59 out 59h, al
4 01A8Ah E4 9A in  al, 9Ah
```

DEĞİŞKENLER

döngü = 2: G/Ç yazma T = T1 A19–A0 = 00059h

BHE' = 0 A0 = 1 D15–D8 = – D7–D0 = – M/IO' = 0

ALE = 1 RD' = 1 WR' = 1 DT/R' = 1 DEN' = 1

AL = 12h

Basitleştirilmiş sıralı model: gerçekte BIU, kuyrukta yer açıldıkça kodu **önden getirir**; döngü 3 döngü 2'den önce de olabilir. Her döngünün T1–T4 ayrıtısı: [Yazma](#) ve [Okuma Yol Döngüsü](#). 8 bitlik birimi yalnız D7–D0'a bağlı olan sistemde tek port adresi kullanılamaz.

Komuttan Yol Döngülerine: OUT ve IN

T2–T4: $WR' = 0$. AL'deki 12h **D15–D8** üzerinden çıkar (tek adres = yüksek banka). Çıkış portu veriyi genellikle WR' yükselirken kilitler.

Adım 4 / 8

```
1 ; CS = 01A7h, AL = 12h
2 ; FA      bayt   komut
▶ 01A88h E6 59 out 59h, al
4 01A8Ah E4 9A in  al, 9Ah
```

DEĞİŞKENLER

döngü = 2: **G/Ç yazma**

T = T2–T4

A19–A0 = **00059h**

BHE' = **0**

A0 = **1**

D15–D8 = 12h

D7–D0 = **–**

M/IO' = **0**

ALE = 0

RD' = **1**

WR' = 0

DT/R' = **1**

DEN' = 0

AL = 12h

Basitleştirilmiş sıralı model: gerçekte BIU, kuyrukta yer açıldıkça kodu **önden getirir**; döngü 3 döngü 2'den önce de olabilir. Her döngünün T1–T4 ayrıntısı: [Yazma](#) ve [Okuma Yol Döngüsü](#). 8 bitlik birimi yalnız D7–D0'a bağlı olan sistemde tek port adresi kullanılamaz.

Komuttan Yol Döngülerine: OUT ve IN

Döngü 3, T1: sonraki komut 01A8Ah'de (çift) → yine tek sözcük getirme. $M/IO' = 1$, $DT/R' = 0$.

Adım 5 / 8

```
1 ; CS = 01A7h, AL = 12h
2 ; FA      bayt   komut
3 01A88h E6 59 out 59h, al
▶ 01A8Ah E4 9A in  al, 9Ah
```

DEĞİŞKENLER

döngü = 3: kod getirme

T = T1

A19-A0 = 01A8Ah

BHE' = 0

A0 = 0

D15-D8 = -

D7-D0 = -

M/IO' = 1

ALE = 1

RD' = 1

WR' = 1

DT/R' = 0

DEN' = 1

AL = 12h

Basitleştirilmiş sıralı model: gerçekte BIU, kuyrukta yer açıldıkça kodu **önden getirir**; döngü 3 döngü 2'den önce de olabilir. Her döngünün T1-T4 ayrıntısı: [Yazma](#) ve [Okuma Yol Döngüsü](#). 8 bitlik birimi yalnız D7-D0'a bağlı olan sistemde tek port adresi kullanılamaz.

Komuttan Yol Döngülerine: OUT ve IN

T2–T4: E4h (IN AL, imm8) D7–D0'dan, 9Ah (port) D15–D8'den okunur.

Adım 6 / 8

```
1 ; CS = 01A7h, AL = 12h
2 ; FA      bayt   komut
3 01A88h  E6 59  out 59h, al
▶ 01A8Ah  E4 9A  in  al, 9Ah
```

DEĞİŞKENLER

döngü = 3: kod getirme

T = T2–T4

A19–A0 = 01A8Ah

BHE' = 0

A0 = 0

D15–D8 = 9Ah

D7–D0 = E4h

M/IO' = 1

ALE = 0

RD' = 0

WR' = 1

DT/R' = 0

DEN' = 0

AL = 12h

Basitleştirilmiş sıralı model: gerçekte BIU, kuyrukta yer açıldıkça kodu **önden getirir**; döngü 3 döngü 2'den önce de olabilir. Her döngünün T1–T4 ayrıntısı: [Yazma](#) ve [Okuma Yol Döngüsü](#). 8 bitlik birimi yalnız D7–D0'a bağlı olan sistemde tek port adresi kullanılamaz.

Komuttan Yol Döngülerine: OUT ve IN

Döngü 4, T1: G/Ç adresi 009Ah **çift** → BHE' = 1, A0 = 0: yalnız düşük banka. Okuma: DT/R' = 0.

Adım 7 / 8

```
1 ; CS = 01A7h, AL = 12h
2 ; FA      bayt  komut
3 01A88h E6 59 out 59h, al
▶ 01A8Ah E4 9A in  al, 9Ah
```

DEĞİŞKENLER

döngü = 4: G/Ç okuma T = T1 A19-A0 = 0009Ah

BHE' = 1 A0 = 0 D15-D8 = - D7-D0 = - M/IO' = 0

ALE = 1 RD' = 1 WR' = 1 DT/R' = 0 DEN' = 1

AL = 12h

Basitleştirilmiş sıralı model: gerçekte BIU, kuyrukta yer açıldıkça kodu **önden getirir**; döngü 3 döngü 2'den önce de olabilir. Her döngünün T1–T4 ayrıtısı: [Yazma](#) ve [Okuma Yol Döngüsü](#). 8 bitlik birimi yalnız D7–D0'a bağlı olan sistemde tek port adresi kullanılamaz.

Komuttan Yol Döngülerine: OUT ve IN

T2–T4: RD' = 0; port 3Ch değerini **D7–D0**'a koyar, T4 başında AL'ye alınır. Yol etkinliği: 4 döngü × 4 T = 16 saat turu (Tw yok).

Adım 8 / 8

```
1 ; CS = 01A7h, AL = 12h
2 ; FA      bayt   komut
3 01A88h E6 59 out 59h, al
▶ 01A8Ah E4 9A in al, 9Ah
```

DEĞİŞKENLER

döngü = 4: **G/Ç okuma** T = T2–T4 A19–A0 = 0009Ah

BHE' = 1 A0 = 0 D15–D8 = – **D7–D0 = 3Ch** M/IO' = 0

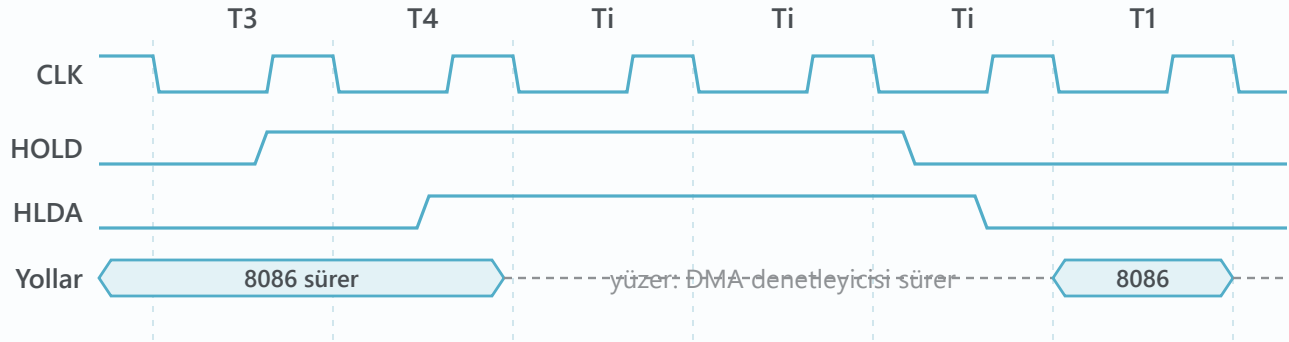
ALE = 0 RD' = 0 WR' = 1 DT/R' = 0 DEN' = 0

AL = 3Ch

Basitleştirilmiş sıralı model: gerçekte BIU, kuyrukta yer açıldıkça kodu **önden getirir**; döngü 3 döngü 2'den önce de olabilir. Her döngünün T1–T4 ayrıntısı: [Yazma](#) ve [Okuma Yol Döngüsü](#). 8 bitlik birimi yalnız D7–D0'a bağlı olan sistemde tek port adresi kullanılamaz.

KONTROL PİNLERİ

HOLD / HLDA (Minimum Mod)

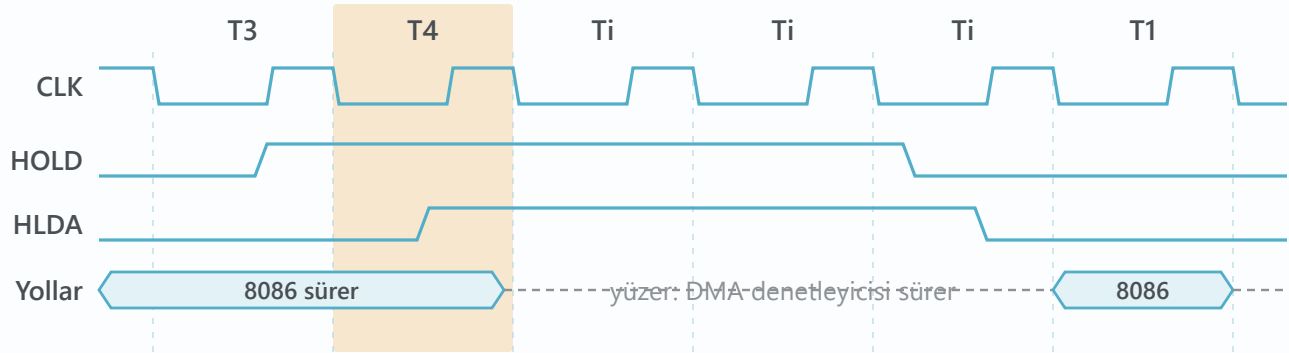


Başka bir yol yöneticisi (ör. 8237 DMA denetleyicisi) yerel yolu kullanmak istiyor.

Adım 1 / 5

Veri sayfasına göre HLDA, T4 veya **Ti** (boşta) turunun ortasında verilir.

HOLD / HLDA (Minimum Mod)



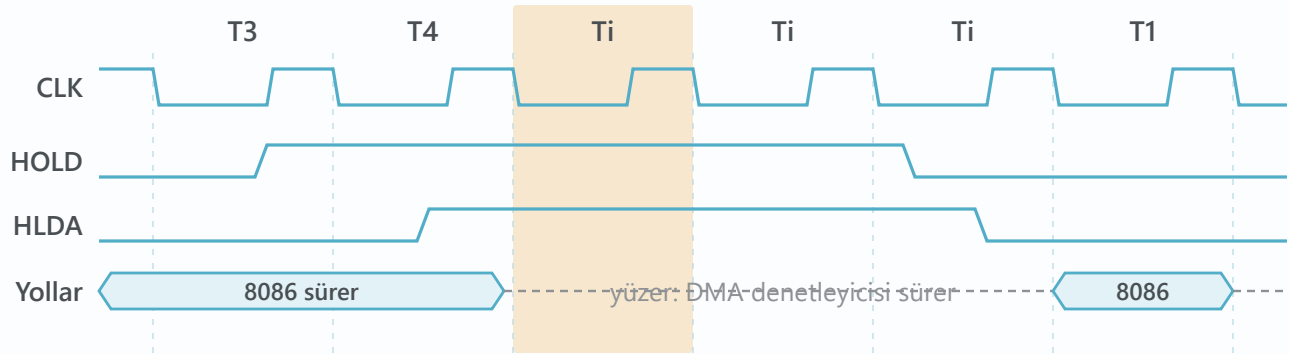
HOLD = 1 (aktif yüksek). İşlemci içinde bulunduğu yol döngüsünü bitirir.

Adım 2 / 5

Veri sayfasına göre HLDA, T4 veya **Ti** (boşta) turunun ortasında verilir.

KONTROL PİNLERİ

HOLD / HLDA (Minimum Mod)

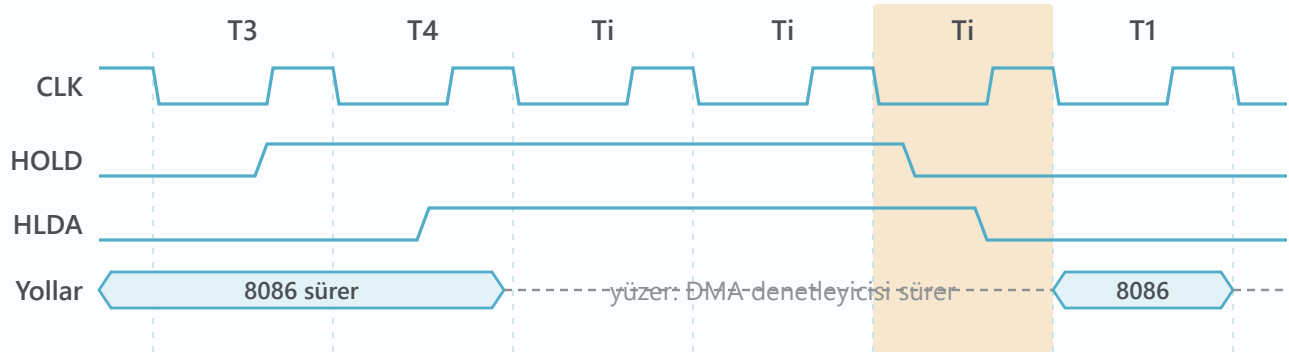


T4 veya T_i 'nin ortasında **HLDA** = 1 ile onay verilir; adres, veri ve kontrol pinleri yüzer duruma geçer.

Adım 3 / 5

Veri sayfasına göre HLDA, T4 veya T_i (boşta) turunun ortasında verilir.

HOLD / HLDA (Minimum Mod)

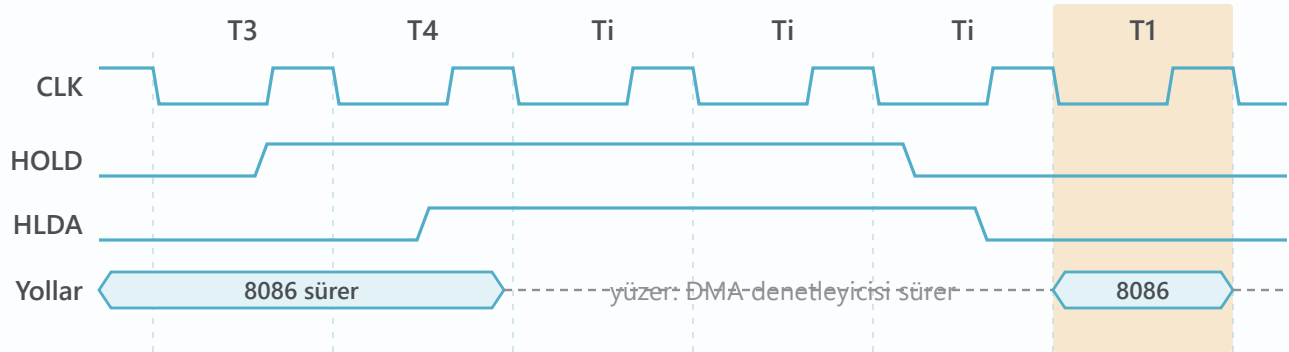


DMA denetleyicisi yolu kullanır. 8086 kuyruktaki komutları yürütmeye devam edebilir, yol gerektiren işte bekler.

Adım 4 / 5

Veri sayfasına göre HLDA, T4 veya **Ti** (boşta) turunun ortasında verilir.

HOLD / HLDA (Minimum Mod)



HOLD = 0 → işlemci HLDA'yı düşürür ve yolu geri alarak yeni bir döngüye başlar.

Adım 5 / 5

Veri sayfasına göre HLDA, T4 veya **Ti** (boşta) turunun ortasında verilir.

RQ'/GT0', RQ'/GT1' ve LOCK'

RQ'/GT · İstek / Bağış (pin 30, 31; maks. mod)

- Diğer yerel yol yöneticileri (ör. 8087) tarafından kullanılır.
- İşlemciyi döngü sonunda yerel yolu **bırakmaya** zorlar.
- Her pin **çift yönlüdür**: tek bir hat üzerinde darbe dizisi → istek (RQ), bağış (GT), bırakma.
- **RQ'/GT0'**, RQ'/GT1' pininden daha yüksek önceliklidir.
- Minimum moddaki HOLD/HLDA'nın maksimum mod karşılığıdır.

LOCK' · Kilit (pin 29; maks. mod)

- Aktif düşük çıkış.
- Aktifken diğer yol yöneticilerinin sistem yolunu almasına **izin verilmez**.
- **LOCK** önekli komut boyunca (komut tamamlanana kadar) aktiftir.
- Çok işlemcili sistemlerde semafor için: **LOCK XCHG AL, [BX]** ile oku-değiştir-yaz bölünmez olur.
- INTA döngüleri sırasında da otomatik olarak aktiftir.

Kod İzleme: LOCK XCHG ile Bölünmez Erişim

```
1 org 100h
```

```
► mov bx, offset bayrak
```

```
3 bekle: mov al, 1
```

```
4 lock xchg al, [bx] ; oku + yaz
```

```
5 cmp al, 0 ; boş muydu?
```

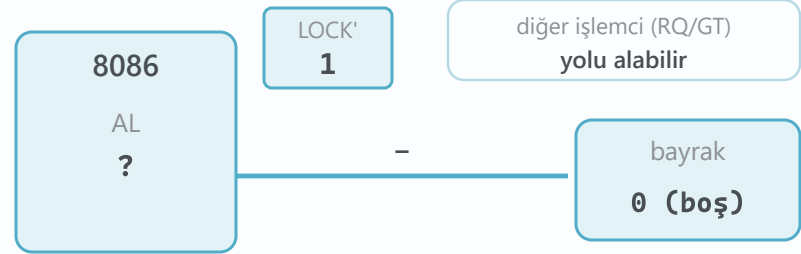
```
6 jne bekle ; doluydu dene
```

```
7 ; ... kritik bölge ...
```

```
8 mov byte ptr [bx], 0 ; bırak
```

```
9 ret
```

```
10 bayrak db 0
```



kritik bölge: —

Ortak bellekteki **bayrak** (010Fh) iki işlemcinin paylaştığı bir semafordur: 0 = boş, 1 = dolu. Şu an boş.

Adım 1 / 8

DEĞİŞKENLER

LOCK' = 1

bayrak = 0

AL = ?

Kod İzleme: LOCK XCHG ile Bölünmez Erişim

```
1 org 100h
2 mov bx, offset bayrak
▶ bekle: mov al, 1
4 lock xchg al, [bx] ; oku + yaz
5 cmp al, 0 ; boş muydu?
6 jne bekle ; doluydu dene
7 ; ... kritik bölge ...
8 mov byte ptr [bx], 0 ; bırak
9 ret
10 bayrak db 0
```



AL = 1: bayrağa yazılacak “dolmuş” değeri.

Adım 2 / 8

DEĞİŞKENLER

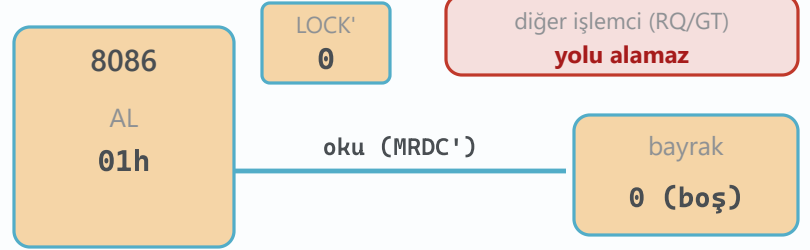
LOCK' = 1

bayrak = 0

AL = 01h

Kod İzleme: LOCK XCHG ile Bölünmez Erişim

```
1 org 100h
2 mov bx, offset bayrak
3 bekle: mov al, 1
4 ► lock xchg al, [bx] ; oku + yaz
5 cmp al, 0 ; boş muydu?
6 jne bekle ; doluydu dene
7 ; ... kritik bölge ...
8 mov byte ptr [bx], 0 ; bırak
9 ret
10 bayrak db 0
```



kritik bölge: —

`lock xchg` başladı: **LOCK' = 0**. 1. yol döngüsü bayrağı okur (0). Diğer yol yöneticisinin RQ/GT isteği bu sürede kabul edilmez.

Adım 3 / 8

DEĞİŞKENLER

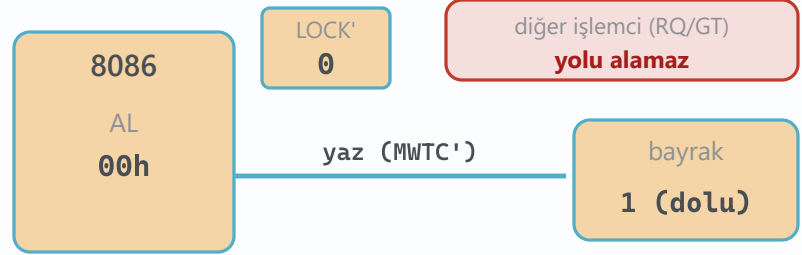
LOCK' = 0

bayrak = 0

AL = 01h

Kod İzleme: LOCK XCHG ile Bölünmez Erişim

```
1 org 100h
2 mov bx, offset bayrak
3 bekle: mov al, 1
4 ► lock xchg al, [bx] ; oku + yaz
5 cmp al, 0 ; boş muydu?
6 jne bekle ; doluydu dene
7 ; ... kritik bölge ...
8 mov byte ptr [bx], 0 ; bırak
9 ret
10 bayrak db 0
```



kritik bölge: bizde

2. yol döngüsü AL'deki 1'i bayrağa yazar; AL eski değeri (0) aldı. Okuma ile yazma arasına başka işlemci **giremedi**.

Adım 4 / 8

DEĞİŞKENLER

LOCK' = 0

bayrak = 1

AL = 00h

Kod İzleme: LOCK XCHG ile Bölünmez Erişim

```
1 org 100h
2 mov bx, offset bayrak
3 bekle: mov al, 1
4 lock xchg al, [bx] ; oku + yaz
5 ▶ cmp al, 0 ; boş muydu?
6 jne bekle ; doluysa dene
7 ; ... kritik bölge ...
8 mov byte ptr [bx], 0 ; bırak
9 ret
10 bayrak db 0
```



XCHG bitti, **LOCK' = 1**: yol yeniden paylaşılabilir. `cmp al, 0` : eski değer 0 → ZF = 1.

Adım 5 / 8

DEĞİŞKENLER

LOCK' = 1

bayrak = 1

AL = 00h

Kod İzleme: LOCK XCHG ile Bölünmez Erişim

```
1 org 100h
2 mov bx, offset bayrak
3 bekle: mov al, 1
4 lock xchg al, [bx] ; oku + yaz
5 cmp al, 0 ; boş muydu?
▶ jne bekle ; doluysa dene
7 ; ... kritik bölge ...
8 mov byte ptr [bx], 0 ; bırak
9 ret
10 bayrak db 0
```



jne atlamaz: bayrak boştu ve artık bizim. Diğer işlemci denese, xchg ona 1 döndürür ve beklemeye devam eder.

Adım 6 / 8

DEĞİŞKENLER

LOCK' = 1

bayrak = 1

AL = 00h

Kod İzleme: LOCK XCHG ile Bölünmez Erişim

```
1 org 100h
2 mov bx, offset bayrak
3 bekle: mov al, 1
4 lock xchg al, [bx] ; oku + yaz
5 cmp al, 0 ; boş muydu?
6 jne bekle ; doluysa dene
7 ; ... kritik bölge ...
▶ mov byte ptr [bx], 0 ; bırak
9 ret
10 bayrak db 0
```



Kritik bölge bitti: bayrak = 0 yazılarak semafor serbest bırakılır.

Adım 7 / 8

DEĞİŞKENLER

LOCK' = 1

bayrak = 0

AL = 00h

Kod İzleme: LOCK XCHG ile Bölünmez Erişim

```
1 org 100h
2 mov bx, offset bayrak
3 bekle: mov al, 1
4 lock xchg al, [bx] ; oku + yaz
5 cmp al, 0 ; boş muydu?
6 jne bekle ; doluydu dene
7 ; ... kritik bölge ...
8 mov byte ptr [bx], 0 ; bırak
9 ► ret
10 bayrak db 0
```



LOCK öneki olmasaydı, diğer işlemci okuma ve yazma döngüleri arasında yolu alıp aynı 0'ı okuyabilir; iki işlemci aynı anda kritik bölgeye girerdi.

Adım 8 / 8

DEĞİŞKENLER

LOCK' = 1

bayrak = 0

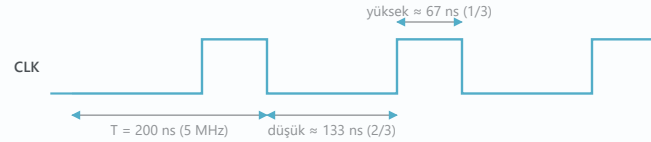
AL = 00h

TEST' ve CLK

TEST' (pin 23)

- **WAIT** komutu tarafından sınanır.
- TEST' = 0 ise yürütme devam eder; değilse işlemci **boşta** bekler ve pini tekrar sınar.
- Her saat turunun yükselen kenarında içeride senkronlanır.
- Tipik kullanım: 8087'nin BUSY çıkışına bağlanır; **FWAIT** ile 8087'nin işini bitirmesi beklenir.

CLK (pin 19): işlemci ve yol kontrolü için temel zamanlamayı sağlar. %33 görev çevrimli (duty cycle) **asimetrik** kare dalgadır.



8086: 5 MHz, 8086-2: 8 MHz, 8086-1: 10 MHz.

KONTROL PİNLERİ

RESET, VCC ve GND

- **RESET** (pin 21): işlemcinin o anki faaliyetini **hemen** sonlandırır. Sinyal en az **dört saat turu** 1 kalmalıdır (açılışta en az 50 μ s). Düştüğünde işlemci yeniden başlar.
- **VCC** (pin 40): +5 V DC besleme.
- **GND** (pin 1, 20): toprak. İki toprak pini, yüksek akımlı anahtarlama gürültüyü azaltır.

Sıfırlama sonrası	Değer
CS	FFFFh
IP	0000h
DS, SS, ES	0000h
FLAGS	temizlenir (IF = 0: kesmeler kapalı)
Kuyruk	boş
İlk komut adresi	FFFF0h + 0000h = FFFF0h

FFFF0h, 1 MB'ın son 16 baytıdır; buraya ROM (BIOS) yerleştirilir ve genellikle asıl başlangıç koduna uzak bir **JMP** bulunur.

İşlemcinin Yaptığı Adımlar: RESET → İlk Komut

► ; RESET = 1 (≥ 4 saat turu)

```
2 CS ← FFFFh, IP ← 0000h
3 DS, SS, ES ← 0000h
4 FLAGS ← 0000h ; IF = 0
5 kuyruk boşaltılır
6 ; RESET 1 → 0
7 getir FFFF0h ; EA 5B E0 00 F0
8 jmp far F000h:E05Bh
9 getir FE05Bh ; BIOS
```

CS

????

DS/SS/ES

????

IP

????

FLAGS

????

1 MB bellek (üst kısım)

FFFF0h: EA 5B E0 00 F0

FFFFh = son bayt

FE05Bh: BIOS kodu

ROM (BIOS)

RESET pini en az dört saat turu 1 tutuluyor: işlemci o anki işini hemen bırakır. Yazmaçların içeriği belirsizdir.

Adım 1 / 8

DEĞİŞKENLER

CS = ????

IP = ????

DS = ????

FLAGS = ????

İşlemcinin Yaptığı Adımlar: RESET → İlk Komut

1 ; RESET = 1 (≥ 4 saat turu)

► CS ← FFFFh, IP ← 0000h

3 DS, SS, ES ← 0000h

4 FLAGS ← 0000h ; IF = 0

5 kuyruk boşaltılır

6 ; RESET 1 → 0

7 getir FFFF0h ; EA 5B E0 00 F0

8 jmp far F000h:FE05Bh

9 getir FE05Bh ; BIOS

CS

FFFFh

DS/SS/ES

????

IP

0000h

FLAGS

????

1 MB bellek (üst kısım)

FFFF0h: EA 5B E0 00 F0

FFFFh = son bayt

FE05Bh: BIOS kodu

ROM (BIOS)

Donanım **CS = FFFFh**, **IP = 0000h** yükler. Programcı bu değerleri seçmez.

Adım 2 / 8

DEĞİŞKENLER

CS = FFFFh

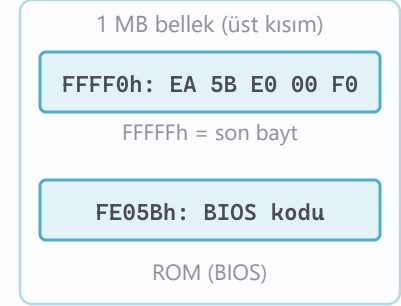
IP = 0000h

DS = ????

FLAGS = ????

İşlemcinin Yaptığı Adımlar: RESET → İlk Komut

```
1 ; RESET = 1 (≥ 4 saat turu)
2 CS ← FFFFh, IP ← 0000h
▶ DS, SS, ES ← 0000h
4 FLAGS ← 0000h ; IF = 0
5 kuyruk boşaltılır
6 ; RESET 1 → 0
7 getir FFFF0h ; EA 5B E0 00 F0
8 jmp far F000h:E05Bh
9 getir FE05Bh ; BIOS
```



Diğer kesim yazmaçları 0000h olur.

Adım 3 / 8

DEĞİŞKENLER

CS = FFFFh

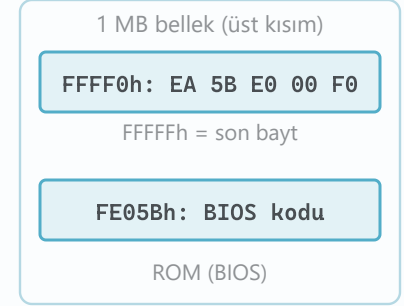
IP = 0000h

DS = 0000h

FLAGS = ????

İşlemcinin Yaptığı Adımlar: RESET → İlk Komut

```
1 ; RESET = 1 (≥ 4 saat turu)
2 CS ← FFFFh, IP ← 0000h
3 DS, SS, ES ← 0000h
▶ 4 FLAGS ← 0000h ; IF = 0
5 kuyruk boşaltılır
6 ; RESET 1 → 0
7 getir FFFF0h ; EA 5B E0 00 F0
8 jmp far F000h:E05Bh
9 getir FE05Bh ; BIOS
```



Bayraklar temizlenir; **IF = 0**: kesmeler kapalı, başlangıç kodu kesilmeden çalışır.

Adım 4 / 8

DEĞİŞKENLER

CS = FFFFh

IP = 0000h

DS = 0000h

FLAGS = 0000h

İşlemcinin Yaptığı Adımlar: RESET → İlk Komut

```
1 ; RESET = 1 (≥ 4 saat turu)
2 CS ← FFFFh, IP ← 0000h
3 DS, SS, ES ← 0000h
4 FLAGS ← 0000h ; IF = 0
```

► kuyruk boşaltılır

► ; RESET 1 → 0

```
7 getir FFFF0h ; EA 5B E0 00 F0
8 jmp far F000h:E05Bh
9 getir FE05Bh ; BIOS
```

CS

FFFFh

DS/SS/ES

0000h

IP

0000h

FLAGS

0000h

1 MB bellek (üst kısım)

FFFF0h: EA 5B E0 00 F0

FFFFh = son bayt

FE05Bh: BIOS kodu

ROM (BIOS)

Kuyruk boş. RESET 0'a inince BIU ilk getirmeyi başlatır.

Adım 5 / 8

DEĞİŞKENLER

CS = FFFFh

IP = 0000h

DS = 0000h

FLAGS = 0000h

İşlemcinin Yaptığı Adımlar: RESET → İlk Komut

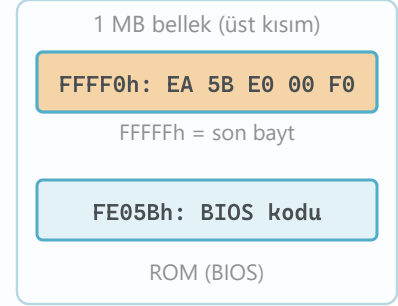
```
1 ; RESET = 1 (≥ 4 saat turu)
2 CS ← FFFFh, IP ← 0000h
3 DS, SS, ES ← 0000h
4 FLAGS ← 0000h ; IF = 0
5 kuyruk boşaltılır
6 ; RESET 1 → 0
```

```
► getir FFFF0h ; EA 5B E0 00 F0
8 jmp far F000h:E05Bh
9 getir FE05Bh ; BIOS
```



adres yolu: FFFF0h

İlk adres: $\text{FFFFh} \times 10\text{h} + 0000\text{h} = \text{FFFF0h}$, 1 MB'ın son 16 baytı. Burada ROM'daki 5 baytlık uzak atlama durur: **EA 5B E0 00 F0**.



Adım 6 / 8

DEĞİŞKENLER

CS = FFFFh

IP = 0000h

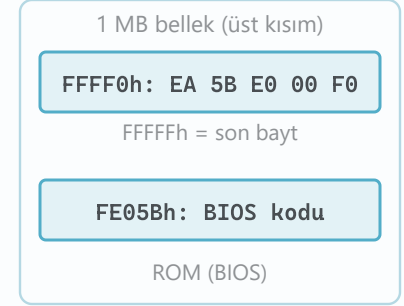
DS = 0000h

FLAGS = 0000h

adres = FFFF0h

İşlemcinin Yaptığı Adımlar: RESET → İlk Komut

```
1 ; RESET = 1 (≥ 4 saat turu)
2 CS ← FFFFh, IP ← 0000h
3 DS, SS, ES ← 0000h
4 FLAGS ← 0000h ; IF = 0
5 kuyruk boşaltılır
6 ; RESET 1 → 0
7 getir FFFF0h ; EA 5B E0 00 F0
8 ► jmp far F000h:E05Bh
9 getir FE05Bh ; BIOS
```



EA = uzak JMP: önce bağlı konum (E05Bh), sonra kesim (F000h), ikisi de little-endian. CS:IP = **F000h:E05Bh**; kuyruk yeniden temizlenir.

Adım 7 / 8

DEĞİŞKENLER

CS = **F000h**

IP = **E05Bh**

DS = **0000h**

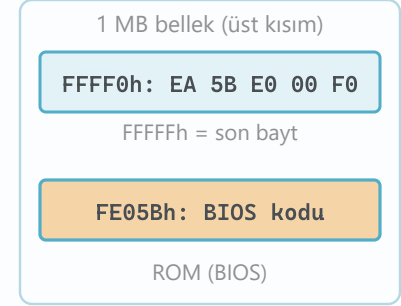
FLAGS = **0000h**

İşlemcinin Yaptığı Adımlar: RESET → İlk Komut

```
1 ; RESET = 1 (≥ 4 saat turu)
2 CS ← FFFFh, IP ← 0000h
3 DS, SS, ES ← 0000h
4 FLAGS ← 0000h ; IF = 0
5 kuyruk boşaltılır
6 ; RESET 1 → 0
7 getir FFFF0h ; EA 5B E0 00 F0
8 jmp far F000h:E05Bh
▶ getir FE05Bh ; BIOS
```



adres yolu: FE05Bh



Yeni adres F0000h + E05Bh = **FE05Bh**: BIOS'un açılış kodu (IBM PC'deki adresi). 16 bayta sığmayan başlangıç kodu buradan sürer.

Adım 8 / 8

DEĞİŞKENLER

CS = **F000h**

IP = **E05Bh**

DS = **0000h**

FLAGS = **0000h**

adres = FE05Bh

8284A Saat Üreteci



OSC = 15 MHz, PCLK = 2,5 MHz (çevre birimleri)

- Kristal frekansını **3'e böler**: 15 MHz → 5 MHz CLK, %33 görev çevrimi.
- READY ve RESET girişlerini saate **senkronlar**; RES' girişindeki Schmitt tetikleyici, RC devresinden gelen yavaş yükselen gerilimi temiz bir RESET'e çevirir.

MN/MX' ve Moda Göre Pinler

Minimum mod pinleri (MN/MX' = 1)

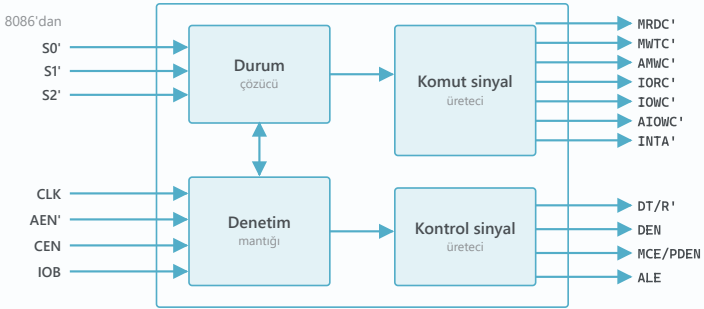
- **ALE** (25): adres kilidi etkinleştirme; T1'de aktif yüksek darbe, 8282/8283 kilitlerini yükler. ALE hiçbir zaman yüzer duruma geçmez.
- **INTA'** (24): kesme kabul.
- **DEN', DT/R', M/IO', WR'**: alıcı-verici ve bellek/G-Ç kontrolü.
- **HOLD, HLDA**: yol devri (DMA).

Maksimum mod pinleri (MN/MX' = 0)

- **S2', S1', S0'**: döngü türü → 8288 kontrol sinyallerini üretir.
- **QS1, QS0**: kuyruk durumu (8087 için).
- **LOCK'**: yolu kilitler.
- **RQ'/GT0', RQ'/GT1'**: yol isteği/bağışı.

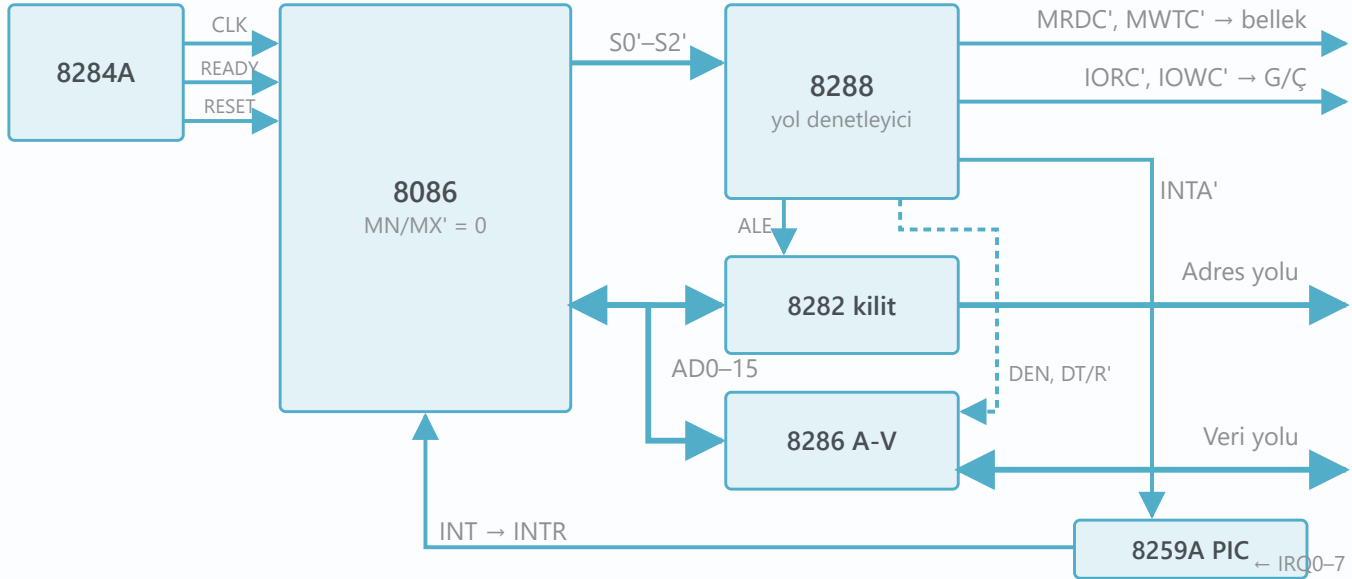
Veri sayfası: "ALE is never floated" — ALE asla yüzer (yüksek empedans) duruma geçmez.

8288 Yol Denetleyicisi



- Maksimum modda 8086'nın S2'–S0' kodunu çözer.
- **Komutlar:** MRDC' bellek oku, MWTC' bellek yaz, IORC'/IOWC' G/Ç oku/yaz, INTA' kesme kabul. "A" ile başlayanlar (AMWC', AIOWC') bir saat önce başlayan "ileri" yazma sinyalleridir.
- **Kontrol:** ALE, DEN, DT/R' artık 8288'den gelir (DEN burada aktif yüksek).
- Pinleri boşalan 8086 bunları QS, LOCK', RQ'/GT' için kullanır.

Maksimum Mod 8086 Sistemi



8284A saati, 8288 komutları ve $ALE/DEN/DT/R'$ sinyallerini, kilitler adres yolunu, alıcı-vericiler veri yolunu, 8259A kesme denetleyicisi de 8 donanım kesmesini ($IRQ0-7$) $INTR$ 'ye bağlar.

INTR, NMI ve INTA'

INTR · Kesme isteği (18)

- **Seviye tetiklemeli**, aktif yüksek giriş.
- Her komutun son saat turunda örneklenir.
- IF = 1 ise kabul edilir (maskelenebilir: CLI ile kapatılır).
- Kabulde işlemci **kesme kabul döngüsüne** girer.
- İçeride senkronlanır.

NMI · Maskelenemeyen kesme (17)

- **Kenar tetiklemeli** ($0 \rightarrow 1$ geçişi).
- Her zaman **tip 2** kesmeye yol açar; vektör tablosunda gösterilen yordam yürütülür.
- Yazılımla (IF) maskelenemez.
- Kesme o anki komutun sonunda başlar.
- Kullanım: güç kesintisi, bellek eşlik hatası.

INTA' · Kesme kabul (24, min.)

- Aktif düşük çıkış.
- Kesme kabul döngüsünün T2, T3 ve Tw'lerinde 0.
- İki ardışık INTA' darbesi üretilir; ikincisinde kesme denetleyicisi **tip numarasını** D0–D7'ye koyar.
- Maks. modda 8288 üretir.

INTR seviye tetiklemelidir; kenar tetiklemeli olan NMI'dır.

Kod İzleme: INTR Ne Zaman Kabul Edilir?

```
1 org 100h
2 cli ; IF = 0
3 mov ax, [sayac] ; INTR = 1 olur
4 inc ax
5 mov [sayac], ax
6 sti ; IF = 1
7 nop ; STI'den sonraki
8 nop ; kesmeden sonra
9 ret
10 sayac dw 41
```

komut

cli

mov

inc

mov

sti

nop

nop

INTR

IF

yok

cli : **IF = 0**. Ortak değişken sayac güncellenirken kesilmek istenmiyor.

Adım 1 / 7

DEĞİŞKENLER

INTR = 0

IF = 0

AX = 0

örnekleme = yok

INTR her komutun son saat turunda örneklenir; kesme isteği örnek bir senaryodur.

Kod İzleme: INTR Ne Zaman Kabul Edilir?

```
1 org 100h
2 cli                ; IF = 0
3 ► mov ax, [sayac]  ; INTR = 1 olur
4 inc ax
5 mov [sayac], ax
6 sti                ; IF = 1
7 nop                ; STI'den sonraki
8 nop                ; kesmeden sonra
9 ret
10 sayac dw 41
```

komut

cli

mov

inc

mov

sti

nop

nop

INTR

IF

yok say

Bu komut sırasında 8259A **INTR = 1** yaptı. Komutun son turunda örneklenir ama IF = 0 → kabul edilmez; INTR seviye tetiklemeli olduğu için 1 kalır.

Adım 2 / 7

DEĞİŞKENLER

INTR = 1

IF = 0

AX = 41

örnekleme = yok sayıldı

INTR her komutun son saat turunda örneklenir; kesme isteği örnek bir senaryodur.

Kod İzleme: INTR Ne Zaman Kabul Edilir?

```
1 org 100h
2 cli                ; IF = 0
3 mov ax, [sayac]    ; INTR = 1 olur
4 ► inc ax
5 mov [sayac], ax
6 sti                ; IF = 1
7 nop                ; STI'den sonraki
8 nop                ; kesmeden sonra
9 ret
10 sayac dw 41
```

komut

cli

mov

inc

mov

sti

nop

nop

INTR

IF

yok say

AX = 42. INTR hâlâ bekliyor.

Adım 3 / 7

DEĞİŞKENLER

INTR = 1

IF = 0

AX = 42

örnekleme = yok sayıldı

INTR her komutun son saat turunda örneklenir; kesme isteği örnek bir senaryodur.

Kod İzleme: INTR Ne Zaman Kabul Edilir?

```
1 org 100h
2 cli                ; IF = 0
3 mov ax, [sayac]    ; INTR = 1 olur
4 inc ax
5 ▶ mov [sayac], ax
6 sti                ; IF = 1
7 nop                ; STI'den sonraki
8 nop                ; kesmeden sonra
9 ret
10 sayac dw 41
```

komut

cli

mov

inc

mov

sti

nop

nop

INTR

IF

yok say

sayac = **42** yazıldı; oku-değiştir-yaz bölünmeden tamamlandı.

Adım 4 / 7

DEĞİŞKENLER

INTR = 1

IF = 0

AX = 42

örnekleme = **yok sayıldı**

INTR her komutun son saat turunda örneklenir; kesme isteği örnek bir senaryodur.

Kod İzleme: INTR Ne Zaman Kabul Edilir?

```
1 org 100h
2 cli                ; IF = 0
3 mov ax, [sayac]    ; INTR = 1 olur
4 inc ax
5 mov [sayac], ax
6 ► sti              ; IF = 1
7 nop                ; STI'den sonraki
8 nop                ; kesmeden sonra
9 ret
10 sayac dw 41
```

komut

cli

mov

inc

mov

sti

nop

nop

INTR

IF

ertelendi

sti : IF = 1. 8086, STI'den hemen sonra kesme kabul etmez; **bir komut daha** yürütür.

Adım 5 / 7

DEĞİŞKENLER

INTR = 1

IF = 1

AX = 42

örnekleme = ertelendi

INTR her komutun son saat turunda örneklenir; kesme isteği örnek bir senaryodur.

Kod İzleme: INTR Ne Zaman Kabul Edilir?

```
1 org 100h
2 cli                ; IF = 0
3 mov ax, [sayac]    ; INTR = 1 olur
4 inc ax
5 mov [sayac], ax
6 sti                ; IF = 1
7 nop                ; STI'den sonraki
8 nop                ; kesmeden sonra
9 ret
10 sayac dw 41
```

komut

cli

mov

inc

mov

sti

nop

nop

INTR

IF

KABUL

STI'yi izleyen komut bitti: INTR = 1 ve IF = 1 → iki INTA döngüsü, bayraklar/CS/IP yığına, IF = 0, kesme yordamına gidilir.

Adım 6 / 7

DEĞİŞKENLER

INTR = 1

IF = 1

AX = 42

örnekleme = KABUL

INTR her komutun son saat turunda örneklenir; kesme isteği örnek bir senaryodur.

Kod İzleme: INTR Ne Zaman Kabul Edilir?

```
1 org 100h
2 cli                ; IF = 0
3 mov ax, [sayac]    ; INTR = 1 olur
4 inc ax
5 mov [sayac], ax
6 sti                ; IF = 1
7 nop                ; STI'den sonraki
8 ▶ nop              ; kesmeden sonra
9 ret
10 sayac dw 41
```

komut

cli

mov

inc

mov

sti

nop

nop

INTR

IF

yok

8259A, INTA döngülerinden sonra INTR'yi düşürdü; yordam IRET ile döndü (IF yeniden 1), program ikinci **nop** 'tan sürer. NMI olsaydı IF'ye bakılmadan, 3. satırdan sonra kabul edilirdi.

Adım 7 / 7

DEĞİŞKENLER

INTR = 0

IF = 1

AX = 42

örnekleme = yok

INTR her komutun son saat turunda örneklenir; kesme isteği örnek bir senaryodur.

Kesme Vektör Tablosu

3FCh	Tip 255	kullanıcı
...	...	
0080h	Tip 32	kullanıcı (32–255)
007Ch	Tip 31	Intel'e ayrılmış
...	5–31	ayrılmış
0014h	Tip 5	ayrılmış
0010h	Tip 4	taşma (INTO)
000Ch	Tip 3	kesme noktası (INT 3)
0008h	Tip 2	NMI pini
0004h	Tip 1	tek adım (TF)
0000h	Tip 0	sıfıra bölme

0000:0000h – 0000:03FFh (ilk 1 KB)

256 giriş × 4 bayt = 1024 bayt

Tip n'nin adresi = $n \times 4$

IP düşük	IP yüksek	CS düşük	CS yüksek
+0	+1	+2	+3

İlk iki bayt: yordamın bağıl konumu (IP)

Son iki bayt: yordamın kesimi (CS)

IBM PC'de BIOS/DOS ayrılmış alanı da kullanır:

08h–0Fh donanım IRQ, 10h video, 21h DOS

“Page fault, general protection” gibi girişler 80286/80386'ya aittir; 8086'da yalnızca tip 0–4 tanımlıdır, 5–31 Intel tarafından ayrılmıştır.

Adım Adım: Vektör Arama

INT 21h

0000:0084h	00h	IP düşük
0000:0085h	10h	IP yüksek
0000:0086h	00h	CS düşük
0000:0087h	F0h	CS yüksek

Bellek değerleri örnektir; gerçek adres işletim sistemine göre değişir.

Yazılım kesmesi **INT 21h** (DOS hizmetleri) yürütülüyor. Tip numarası komutun içinde.

Adım Adım: Vektör Arama

INT 21h → tip 21h × 4 = 84h

0000:0084h	00h	IP düşük
0000:0085h	10h	IP yüksek
0000:0086h	00h	CS düşük
0000:0087h	F0h	CS yüksek

Bellek değerleri örnektir; gerçek adres işletim sistemine göre değişir.

Vektörün adresi: **tip × 4** = 21h × 4 = **84h**. Tablo 0000h kesimindedir.

Adım Adım: Vektör Arama

INT 21h → tip 21h × 4 = 84h

0000:0084h **00h** IP düşük

0000:0085h **10h** IP yüksek

0000:0086h **00h** CS düşük

0000:0087h **F0h** CS yüksek

IP ← 1000h

(little-endian: 85h'deki 10 yüksek bayt)

Bellek değerleri örnektir; gerçek adres işletim sistemine göre değişir.

84h–85h: yordamın bağlı konumu. Little-endian → IP = **1000h**.

Adım Adım: Vektör Arama

INT 21h → tip 21h × 4 = 84h

0000:0084h **00h** IP düşük

0000:0085h **10h** IP yüksek

0000:0086h **00h** CS düşük

0000:0087h **F0h** CS yüksek

IP ← 1000h

(little-endian: 85h'deki 10 yüksek bayt)

CS ← F000h

Bellek değerleri örnektir; gerçek adres işletim sistemine göre değişir.

86h–87h: yordamın kesimi → CS = **F000h**.

Adım Adım: Vektör Arama

INT 21h → tip 21h × 4 = 84h

0000:0084h	00h	IP düşük
0000:0085h	10h	IP yüksek
0000:0086h	00h	CS düşük
0000:0087h	F0h	CS yüksek

IP ← 1000h

(little-endian: 85h'deki 10 yüksek bayt)

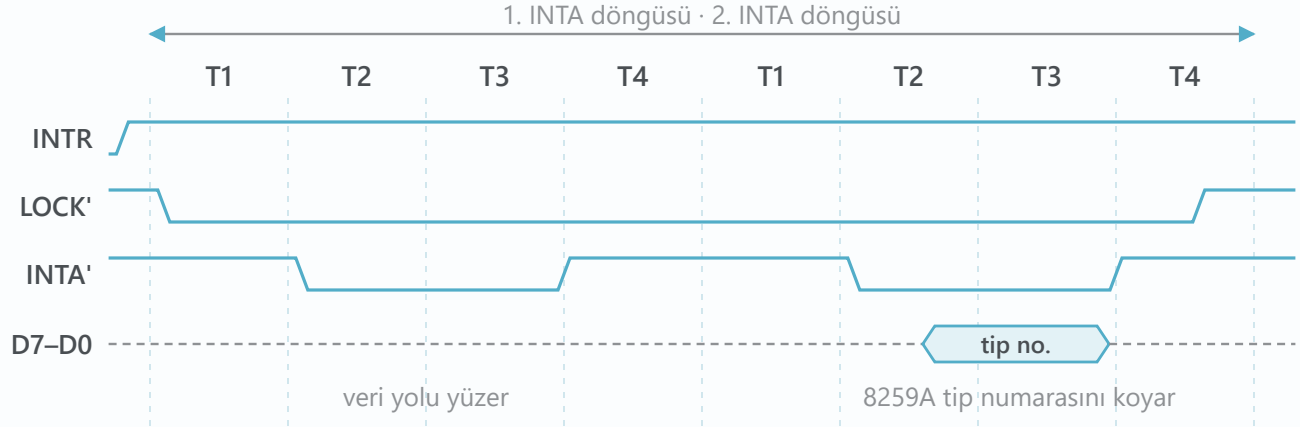
CS ← F000h

Yordam: F000:1000h = F1000h

Bellek değerleri örnektir; gerçek adres işletim sistemine göre değişir.

İşlemci (bayraklar, CS, IP yığına atıldıktan sonra) **F000:1000h** = F1000h adresine dallanır.

Donanım Kesmesi: INTA Döngüleri



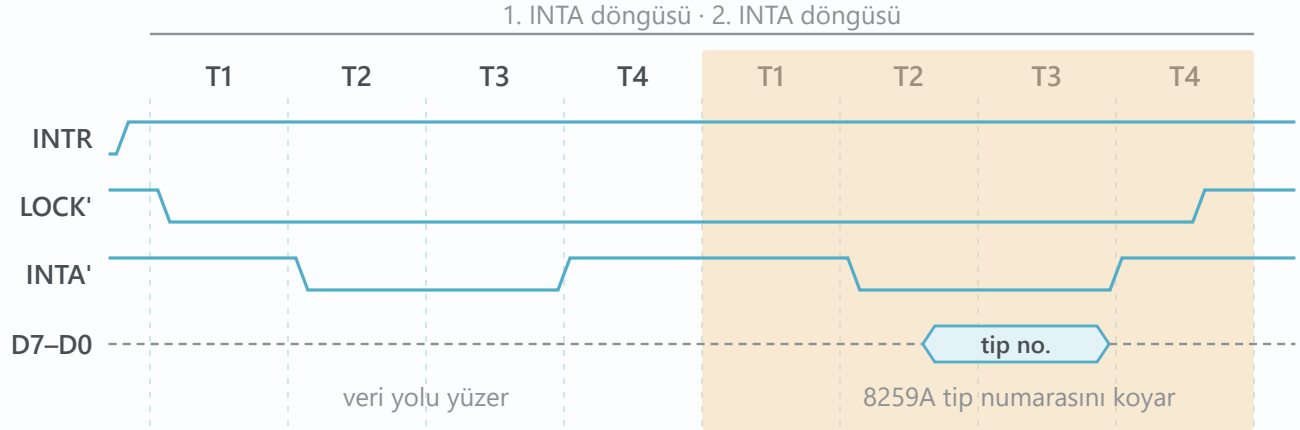
INTR = 1 ve IF = 1 → o anki komut bitince işlemci iki **kesme kabul döngüsü** çalıştırır.

Donanım Kesmesi: INTA Döngüleri



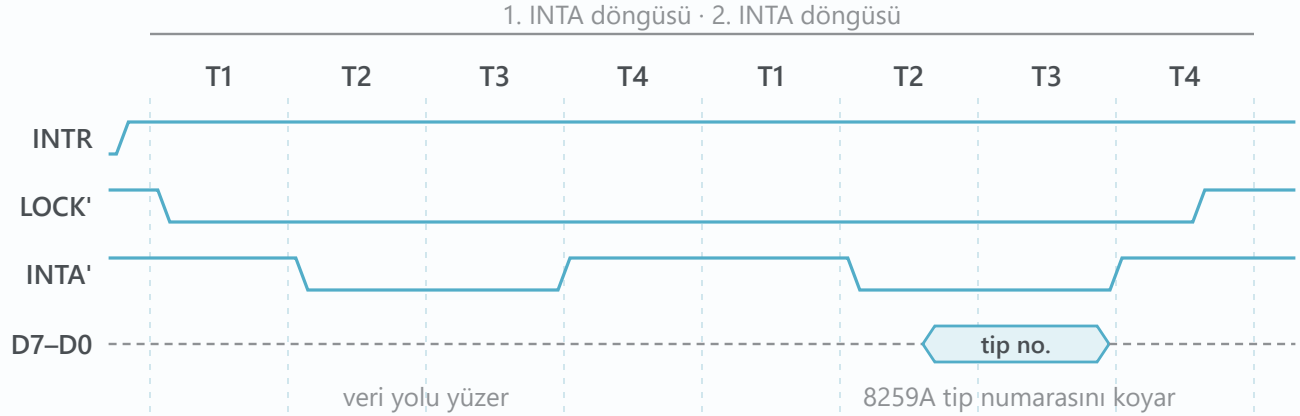
1. INTA' darbesi: kesme denetleyicisine (8259A) "isteğin kabul edildi" bildirilir. Veri yolu yüzer kalır.

Donanım Kesmesi: INTA Döngüleri



2. INTA' darbesi: 8259A, kesme **tip numarasını** (ör. IRQ0 → 08h) D7-D0'a koyar; işlemci okur.

Donanım Kesmesi: INTA Döngüleri



Maks. modda **LOCK'** iki döngü boyunca aktiftir: arada başka bir yöneticinin yolu alması engellenir.

Kesme Yanıt Sırası

Ana program çalışıyor; SS:SP = 0030:0100h olsun. INTR = 1 geldi, IF = 1 → komutun sonunda kabul.

Adım 1 / 9

► ; o anki komut biter (CS:IP = 1000:0120h)

```
2 INTA döngüsü x2      ; tip = 08h
3 PUSHF                ; bayraklar yığına
4 IF ← 0, TF ← 0
5 PUSH CS
6 PUSH IP
7 IP ← [0000:0020h]     ; 08h x 4
8 CS ← [0000:0022h]
9 ; ... kesme yordamı ...
10 IRET                ; IP, CS, FLAGS geri
```

DEĞİŞKENLER

CS = 1000h

IP = 0120h

SP = 0100h

IF = 1

TF = 0

tip = -

ÇAĞRI YIĞINI

boş

Kesme Yanıt Sırası

İki INTA döngüsü; 8259A tip **08h**'yi verir (IRQ0, zamanlayıcı).

Adım 2 / 9

1 ; o anki komut biter (CS:IP = 1000:0120h)

► **INTA döngüsü x2** ; tip = 08h

3 PUSHF ; bayraklar yığına

4 IF ← 0, TF ← 0

5 PUSH CS

6 PUSH IP

7 IP ← [0000:0020h] ; 08h x 4

8 CS ← [0000:0022h]

9 ; ... kesme yordamı ...

10 IRET ; IP, CS, FLAGS geri

DEĞİŞKENLER

CS = 1000h

IP = 0120h

SP = 0100h

TF = 0

IF = 1

tip = 08h

ÇAĞRI YIĞINI

boş

Kesme Yanıt Sırası

PUSHF: $SP - 2 = 00FEh$, bayrak yazmacı yığına.

Adım 3 / 9

```
1 ; o anki komut biter (CS:IP = 1000:0120h)
2 INTA döngüsü x2      ; tip = 08h
▶ 3 PUSHF              ; bayraklar yığına
4 IF ← 0, TF ← 0
5 PUSH CS
6 PUSH IP
7 IP ← [0000:0020h]    ; 08h x 4
8 CS ← [0000:0022h]
9 ; ... kesme yordamı ...
10 IRET                ; IP, CS, FLAGS geri
```

DEĞİŞKENLER

CS = 1000h

IP = 0120h

SP = 00FEh

IF = 1

TF = 0

tip = 08h

ÇAĞRI YIĞINI

FLAGS (IF=1)

Kesme Yanıt Sırası

IF = 0 ve TF = 0: yordam başka INTR ile kesilmesin, adım adım çalışmasın.

Adım 4 / 9

```
1 ; o anki komut biter (CS:IP = 1000:0120h)
2 INTA döngüsü x2      ; tip = 08h
3 PUSHF                ; bayraklar yığına
4 ► IF ← 0, TF ← 0
5 PUSH CS
6 PUSH IP
7 IP ← [0000:0020h]    ; 08h x 4
8 CS ← [0000:0022h]
9 ; ... kesme yordamı ...
10 IRET                ; IP, CS, FLAGS geri
```

DEĞİŞKENLER

CS = 1000h

IP = 0120h

SP = 00FEh

TF = 0

IF = 0

tip = 08h

ÇAĞRI YIĞINI

FLAGS (IF=1)

Kesme Yanıt Sırası

PUSH CS: SP = 00FCh.

Adım 5 / 9

```
1 ; o anki komut biter (CS:IP = 1000:0120h)
2 INTA döngüsü x2      ; tip = 08h
3 PUSHF                ; bayraklar yığına
4 IF ← 0, TF ← 0
5 ► PUSH CS
6 PUSH IP
7 IP ← [0000:0020h]    ; 08h x 4
8 CS ← [0000:0022h]
9 ; ... kesme yordamı ...
10 IRET                ; IP, CS, FLAGS geri
```

DEĞİŞKENLER

CS = 1000h

IP = 0120h

SP = 00FCh

IF = 0

TF = 0

tip = 08h

ÇAĞRI YIĞINI

CS=1000h

FLAGS (IF=1)

Kesme Yanıt Sırası

PUSH IP: SP = 00FAh. Dönüş adresi 1000:0120h güvende.

Adım 6 / 9

```
1 ; o anki komut biter (CS:IP = 1000:0120h)
2 INTA döngüsü x2      ; tip = 08h
3 PUSHF                ; bayraklar yığına
4 IF ← 0, TF ← 0
5 PUSH CS
6 ► PUSH IP
7 IP ← [0000:0020h]    ; 08h x 4
8 CS ← [0000:0022h]
9 ; ... kesme yordamı ...
10 IRET                ; IP, CS, FLAGS geri
```

DEĞİŞKENLER

CS = 1000h

IP = 0120h

SP = 00FAh

IF = 0

TF = 0

tip = 08h

ÇAĞRI YIĞINI

IP=0120h

CS=1000h

FLAGS (IF=1)

Kesme Yanıt Sırası

IP $\leftarrow$ 08h $\times$ 4 = 20h adresindeki sözcük (örnek: **0100h**).

Adım 7 / 9

```
1 ; o anki komut biter (CS:IP = 1000:0120h)
2 INTA döngüsü x2      ; tip = 08h
3 PUSHF                ; bayraklar yığına
4 IF  $\leftarrow$  0, TF  $\leftarrow$  0
5 PUSH CS
6 PUSH IP
7 ► IP  $\leftarrow$  [0000:0020h] ; 08h  $\times$  4
8 CS  $\leftarrow$  [0000:0022h]
9 ; ... kesme yordamı ...
10 IRET                ; IP, CS, FLAGS geri
```

DEĞİŞKENLER

CS = 1000h

IP = 0100h

SP = 00FAh

IF = 0

TF = 0

tip = 08h

ÇAĞRI YIĞINI

IP=0120h

CS=1000h

FLAGS (IF=1)

Kesme Yanıt Sırası

CS ← 22h adresindeki sözcük (örnek: **2000h**). Yordam 2000:0100h = 20100h'de başlar.

Adım 8 / 9

```
1 ; o anki komut biter (CS:IP = 1000:0120h)
2 INTA döngüsü x2      ; tip = 08h
3 PUSHF                ; bayraklar yığına
4 IF ← 0, TF ← 0
5 PUSH CS
6 PUSH IP
7 IP ← [0000:0020h]    ; 08h x 4
8 ► CS ← [0000:0022h]
9 ; ... kesme yordamı ...
10 IRET                ; IP, CS, FLAGS geri
```

DEĞİŞKENLER

CS = **2000h**

IP = **0100h**

SP = **00FAh**

TF = **0**

IF = **0**

tip = **08h**

ÇAĞRI YIĞINI

IP=**0120h**

CS=**1000h**

FLAGS (IF=1)

Kesme Yanıt Sırası

IRET: IP, CS ve FLAGS yığından geri alınır (SP = 0100h). IF yeniden 1, ana program kaldığı yerden sürer.

Adım 9 / 9

```
1 ; o anki komut biter (CS:IP = 1000:0120h)
2 INTA döngüsü x2      ; tip = 08h
3 PUSHF                ; bayraklar yığına
4 IF ← 0, TF ← 0
5 PUSH CS
6 PUSH IP
7 IP ← [0000:0020h]    ; 08h x 4
8 CS ← [0000:0022h]
9 ; ... kesme yordamı ...
▶ IRET                ; IP, CS, FLAGS geri
```

DEĞİŞKENLER

CS = 1000h

IP = 0120h

SP = 0100h

IF = 1

TF = 0

tip = 08h

ÇAĞRI YIĞINI

boş

ÖZET

Özet

Çoklama

AD0–AD15 T1'de adres, sonra veri; A16–A19 T1'de adres, sonra S3–S6. ALE + 8282 kilit adresi tutar.

BHE' + A0

Çift/tek banka seçimi. Hızsız sözcük iki döngü sürer.

Saat/sıfırlama

8284A: $15 \text{ MHz} / 3 = 5 \text{ MHz}$, %33 görev; RESET sonrası FFFF:0000h.

Yol döngüsü

T1 adres, T2 yön/komut, T3 veri, T4 bitiş.
READY = 0 $\rightarrow$ Tw. 5 MHz'de 800 ns.

Modlar

MN/MX' = 1: kontrol sinyallerini 8086 üretir.
0: S2'–S0' $\rightarrow$ 8288.

Kesmeler

INTR seviye, IF ile maskelenir; NMI kenar, tip 2. Vektör = tip $\times$ 4.

ÖZET

Alıştırmalar

1. 8086 8 MHz'de çalışıyor. Bekleme durumu olmayan bir yol döngüsü kaç ns sürer? İki Tw eklenirse?
2. Bir çevre birimine yazılırken (OUT) M/IO', RD', WR', DT/R' pinleri hangi değerleri alır?
3. `MOV [2003h], AX` kaç yol döngüsü gerektirir? Her döngüde BHE' ve A0 ne olur?
4. $S_2' S_1' S_0' = 1\ 0\ 0$ ile $1\ 0\ 1$ arasındaki fark nedir? 8288 her ikisi için hangi komutu üretir?
5. INT 10h yordamının adresi bellekte hangi baytlarda saklanır?
6. Kristal 24 MHz olursa 8284A hangi CLK frekansını üretir? Bu, hangi 8086 sürümü için uygundur?

► İpuçları / cevaplar

ÖZET

Alıştırmalar (2)

7. $T = 125 \text{ ns}$ olan bir sistemde $T_{\text{CLAV}} + T_{\text{DVCL}} + t_{\text{tampon}} = 120 \text{ ns}$ kabul edilirse 450 ns erişim süreli bellek için kaç T_w gerekir?
8. `OUT 9Bh, AL` komutunun G/Ç yazma döngüsünde A7–A0, BHE', A0 ve verinin geçtiği hatlar nedir?

► İpuçları / cevaplar

Kaynaklar

- Intel, *8086 16-Bit HMOS Microprocessor* veri sayfası (231455).
- Intel, *8284A Clock Generator and Driver, 8288 Bus Controller, 8282/8283 Octal Latch* veri sayfaları.
- B. B. Brey, *The Intel Microprocessors*, 8. baskı, Pearson, 2009 (Bölüm 9: 8086/8088 donanım özellikleri).
- D. V. Hall, *Microprocessors and Interfacing*, 2. baskı, McGraw-Hill.
- M. Bodur, *Mikroişlemcilere Giriş: Assembler ile Yazılım ve Arayüz*, EMO, 2014.
- K. Shirriff, 8086 kalıp fotoğrafları ve tersine mühendislik yazıları, righto.com.

Son

Bölüm 2: 8086 Pinler

Sonraki bölüm: Bölüm 2: 8086 Yazmaçlar · [Terim sözlüğü](#)