MİKROİŞLEMCİLER

Doç. Dr. Rıfat Yazıcı

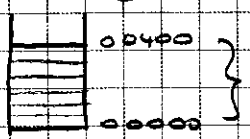
The Indispensable PC Hardware Book

Hans - Peter Messmer

Addison - Wesley

Forth Edition

Bios belleğin en alt bölümüne dış cihazların adreslerini koyar.



İşletim sistemi belleğin en üst kısmını kullanır.

Alttan 2. bölge user program için

" 3. " " " in datası için

" 4. " boş alan olarak

" 5. " string, us. için

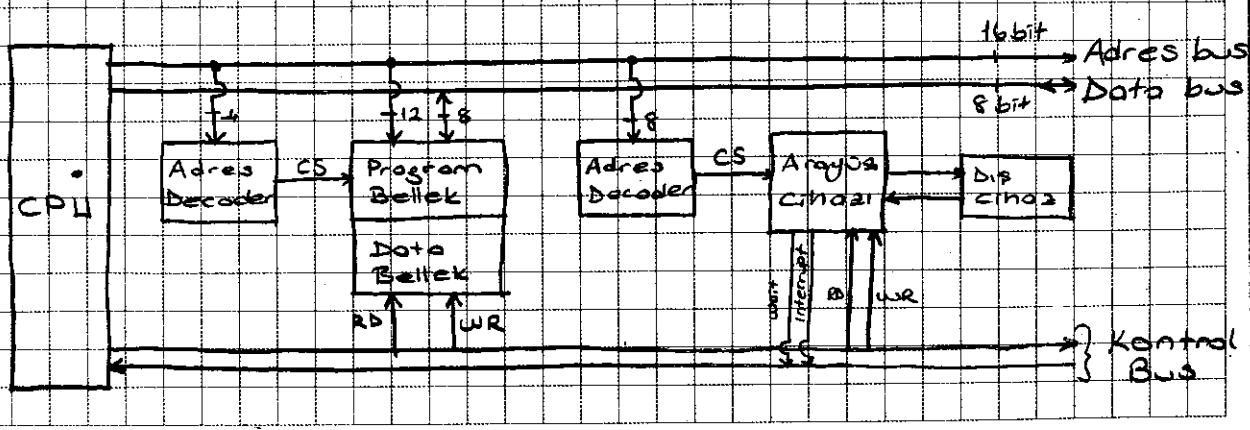
" 6. " boş alan olarak

" 7. " struck, us için

ayrılmıştır.

Struct aşağı doğru, string yukarı doğru büyüdüğü için boş alanları kullanırlar.

Bilgisayarın Genel yapısı



CPU (Central Processing Unit)

8080 CPU → Hemen hemen tek chip halinde yapılmış ilk CPU
→ 8 bitlik
→ 1974 Intel
→ Kontrol birimi dışarıdan bağlanır.

* 6802
  280
  8085
  } 8 bitlik ilk CPU'lar

8080 genel amaçlı kullanılamıyordu. Her işlem için yeni parçalara ihtiyaç duyulduğundan maliyeti çoktu. Daha sonra bu üç CPU çıkarıldı. Bu CPU'lar programlanarak bir parça ile birçok iş yapabildiğinden maliyeti azalttı. Sonra bunlar daha da geliştirilerek PC CPU'ları oluşturuldu.

8086
8088
80286
} 16-bitlik CPU

aritmetik işlem yeteneği iyi değil ↑ 80386
80486
} 32-bitlik CPU

pentium } 32-bitlik CPU

Günümüzde yaygın olarak pentium kullanılıyor. 64 bitlik CPU'lar da var ama uyumlu PC olmadığından ancak çok yüksek bilgisayarlarda kullanılıyor.

Co-precessor

8087 APU } 8086, 8088, 80286, 80386

80387

Bunun sayesinde aritmetik işlem yeteneği artıyordu. Ama bu processor CPU'nun 4-5 kati daha pahalıydı.

80486, pentium } Tek chipten oluşmuş ve aritmetik işlemlerde yetenekli CPU'lar.

Günümüzdeki CPU lerde 20 milyon transistör var. Önceden CPU'larda çok ısı açığa çıkıyordu. Soğutmak sorundu. MOS teknolojisinden sonra bu sorun ortadan kalktı.

Günümüzde CPU'lar çok fazla 'karmaşık emirler içermiyor. Mümkün olduğunca eşit karmaşık-

Likte ve icra süresi aynı emirler kullanılıyor. Eski programları da kolaştırsın diye eski ve yeni yi aynı anda içermesi gerekiyor.

*CPU ancak yarı iletken belleği adresleyebilir. Yarı iletken bellek çeşitleri;

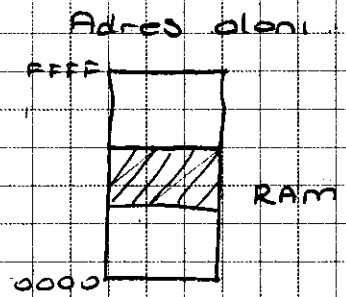
RAM

ROM

PROM

E^2PROM

Günümüzde CPU daki adres busları 36 bittir. (Pentiumlarda). Bu gereksizdir. Çünkü tüm adresleri kullanacak bellek korkunç fiyatta olacaktır. Biz belleği daha sınırlı alırız. Diğer adresler boştur.



Yarı iletken belleklerin adres yapısında oluşturduğu yeri adres decoder tanımlar.

Kontrol bus sayesinde yazma gibi istekler CPU'ya ulaştırılır. Dış cihazların interrupt denen CPU'yu uyarmaya yarayan yolları vardır. Sistemde birçok processor kullanılır. Ama bu processorler slave'dir. CPU (master)'nun izni olmadan çalışamazlar.

Merkezi İşletim Birimi (CPU)

CPU'nun içerisindeki birimler

1. CPU kaydedicileri
2. ALU
3. Kontrol birimi

1. CPU Kaydedicileri (CPU Register)

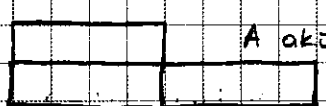
Bu kaydedıcıler CPU'dan CPU'ya değişiklik gösterir. Genelde kullanılanlar şu çeşitlerdir:

a) Akümülatör: Her CPU'da olması gerekir. Data transfonlarını bu register üzerinden gerçekleştirilir. CPU'nun kaç bitlik olacağını belirler. 8 bitlik CPU'da 8 bitlik akümülatör var demektir. 16 " " 16 " " " "

Akümülatör genel olarak A harfiyle gösterilir.

16 bitliklerde AX, 32 bitliklerde EAX olarak gösterilir. Motorolanın 6502 CPU'sunda 2 akümülatör bulunur. Günümüzde bilgisayarlarda daha çok cache bellek kullanılıyor.

b) Genel amaçlı registerler (GR): Her CPU'da bulunmak zorunda değildir. Daha çok Intel ailesinde görülen kaydedici tipidir.

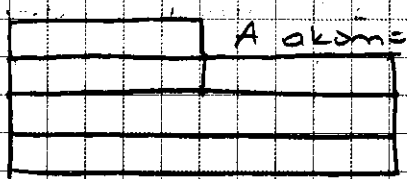


Yazmaçlar, akümülatörlerle aynıdır. Akümülatör ara sonuçları tutmak için kullanılır; for döngüleri için uygun değildir. GR bu işe yarar. GR'den yararlanılan yerler:

$$\left.\begin{matrix} A \\ B \end{matrix}\right) x \quad \left.\begin{matrix} C \\ D \end{matrix}\right) y \quad \Big) x \qquad \underbrace{(A+B)}_{x} \cdot \underbrace{(C+D)}_{y} \rightarrow \text{GR'de saklanır.}$$

CPU aynı anda 2 sayı tutar. Ama sonuçları CPU'da tutulsa hızlı olur. x ve y ara sonuçları burada tutulur. Böylece işlem hızı artar. Bellekte adresleme yapıldığından x ve y ara sonuçları bellekte tutulur, GR'lerde extra adres bilgisine gerek yoktur.

c) Data Counter (DC): Data belleği adreslemeye yarayan, her data bellek erişiminden sonra içeriği kendiliğinden değişemeyen ve her CPU'da bulunması zorunlu olmayan bir kaydedicidir.



H, L intel ailesindeki data counterdır.

İçeriği LOOD DC, emri ile belirlenen bir kaydedicidir.

LOOD DC, Data → Bellekteki veri DC'ye aktarılır. Sıralı dizilerde ilk veri DC'ye aktarıldıktan sonra INC komutu ile diğer veriler de sırayla işlem yapmak için data counter'a aktarılır.

INC, HL

Source index)
Destination index)

d) Program Counter: Program belleği adreslemeye yarayan her program bellek eriş-

miriden sonra içeriği otomatik olarak 1 artan her CPU'da bulunması zorunlu bir kaydedicidir, jmp address → emriyle içeriği istenilen bir değere setlenebilir. Bu adres otomatik olarak HL'ye gider,

e) İndex kaydedici: Data belleği adreslemeye yarayan bir registerdir. LDX Data emriyle içeriği setlenir ve INX emriyle bu içerik güncellenebilir. Diğer datalara erişim daha çok [IR] + offset emrinin kullanıldığı offsetli adresleme sayesinde sağlanır.

LDA A offset,X gibi

Source index bu amaçla kullanılır.

f) Stack pointer: Stack belleği adreslemeye yarayan her CPU'da bulunması zorunlu bir kaydedicidir.

PUSH emriyle datalar stacka iletilirken içeriği güncellenir.

PULL(POP) emriyle datalar stackten çekilirken içeriği güncellenir.

LDS Data ile datayı stacka kayar böylece stack pointer pointerın yerini bildirir.

Bunun dışında INC gibi emirlerle stack artırma,

★Programın kesildiği yeri hatırlamak için o anda-ki değerleri stackta tutulur. Bu bilgileri adres-leyen ise stack registerdır.

g) Emir (Instruction) Register: Program bellekten alınan emir kodunun o emir icrasını tamam-layıncaya kadar CPU'da tutulduğu yerdir.

Emir kodlarının tutulduğu yerdir.

Uzunluğu CPU ile alakalı değildir. Sadece CPU'-nun bulundurduğu emir sayısına bağlıdır. Ne kadar emir o kadar bit.

→Bunun dışında da kullanılması zorunlu ol-mayan registerler vardır.

Register konusunda en zengin CPU'lardan biri Z80 CPU dur.

Z80 deki diğer registerler:

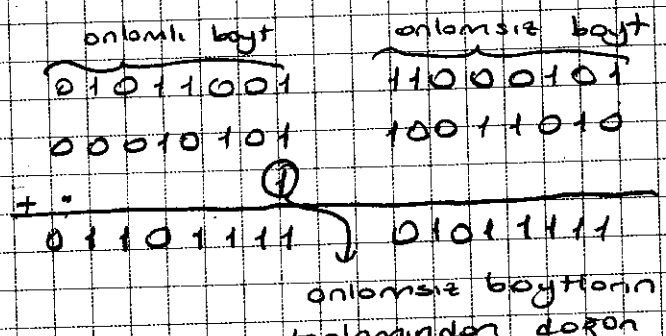
h) Refresh Register: Dinamik belleklerin içindeki bilgilerin yenilenmesi için kullanılan registerdır. PC lerde hiçbir CPU refreshleme işini üstlenmemiş-tir. Zamanında gelişmiş dinamik bellekler kendi kendini refreshliyor ve böylece daha hızlı çalışıyor.

2) Interrupt vektor register: IREQ işareti dışarıdan CPU'ya INT olarak gelir. Bu emirle CPU koşmakta olduğu programı durdurup başka bir programı koşmaya başlar.

Status Flags (Durum Bayrakları)

Bunlar 1 bitlik hücrelerdir.

1. Carry and Intermediary Carry Flags (Elde ve Ara Elde Bayrağı): Çok baytlı sayıların toplanmasında anlamsız baytların toplanmasından doğan eldenin tutulduğu veya çok baytlı veya tek baytlı sayıların kaydırılmasında kaydırma sonucu dışarıya atılan bitin saklandığı yer olarak kullanılan bir bayraktır. Ayrıca bit testinde test edilecek bitin saklandığı yer olarak da kullanılabilir.

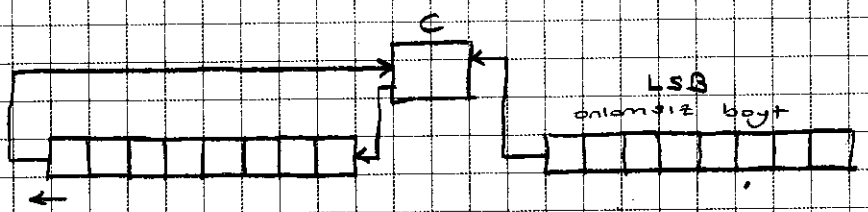


CPU'larda iki tür toplama emri vardır.

ADD A, Data | [Bellek] → carry'i göz ardı eder.

ADC A, Data | [Bellek] → carry'i toplamaya katar

↓ ADD carry ↓ bellekteki bilgi



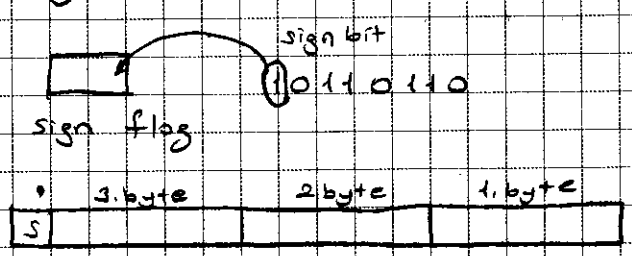
SLA A → Akümülatörün içeriğini sola bir bit kaydırır.

Elde biti kaydırma işleminde bitlerin tutulduğu yer olarak da kullanılabilir.

Bu bayrak her CPU'da vardır.

2- Zero Flag: İşlemin sonucu 0 olduğunda 1'e set lenir. Döngülerde çok sık kullanılır. Her CPU'da bulunur.

3- Sign Flag: İşlem sonucu negatif olduğunda 1'e setlenen bayraktır.



En anlamlı byte'ın en anlamlı biti sign bitidir.

4- Overflow Flag:

+65	0 1 1 0 0 1 0 1
+35	0 0 1 1 1 0 0 0
+90	1 0 0 1 1 1 0 1 = -63

en anlamlı bit

İşaret biti

07	0 0 0 0 1 0 0 0
08	0 0 0 0 0 1 1 1
0E	0 0 0 0 1 1 1 1 = 0E

$C_s = 0$ $C_p = 0$

Bunların ikisi aynıysa sonuçta bir problem yok.

+38	0 0 1 1 1 0 0 0
-24	1 1 0 1 1 1 0 0
+14	0 0 0 1 0 1 0 0 = +14

$C_s = 1$ $C_p = 1$

-38	1 0 1 0 1 0 0 0
-8E	1 0 0 1 0 0 1 0
-66	0 0 1 1 1 0 1 0 = +3A

$C_s = 1$ $C_p = 0$

İşaret bitlerinin toplamı (C_s) ile sayının en anlamlı bitlerinin toplamı aynıysa sorun yok. Farklıysa taşma var.

Overflow = $C_s \oplus C_p$

Hem ustten hem alttan taşmalar bu bayrak sayesinde tespit edilir.

$0,\overset{0,5}{x}\ \overset{0,25}{x}$ ↓

Bilgisayar m.n. algılayabildiğinden daha küçük değerleri 0 olarak tutar, Buna da alttan taşma denir.

5- Parity Flag: Sayının lojik-1 olan bit sayısının tek ya da çift olduğunu gösterir. Genellikle kontrol amaçlı kullanılır. Sayının bitlerinin değişip değişmediğini anlamamızı sağlar. Günümüzdeki bilgisayarlarda hamming code kullanılıyor. Böylece 2 bit değişimi de anlaşılıyor. Ama hamming code daha çok parity bit kullanıyor.

Her CPU'da olmayan bayraklar:

6- Trace flag: Program üzerinde adım adım gidileceği zaman bu bayraktan yararlanılır. (Programın belli yerlerinde hatalar olsun.) Yüksek seviyeli dillerde breaklar yardımıyla hata yerleri tespit edilebilir. Assembly'de ise emir emir koşma yapılır.

7- Interrupt flag: Bir program bu bayrağı setlediğinde CPU kendi işini bırakıp onu kesmeye başlar Tüm CPU'larda yardır. Bu bayrağı setlemek için bir emir vardır; bunu programcı kullanır.

8- Direction flag: Stringin bir yerden başka yere taşınması sırasında buna ihtiyaç duyulur. Datayı aldıktan sonra bir sonraki datanın adresini arttırmak ya da azaltmak için kullanılır. İleri ya da geri otomatik 1 arttırma ya da 1 azalma yapılacağını belirleyen bayraktır.

9- Nested Task flagı: Bir olayın içerisinde başka bir olay cereyan ediyorsa kullanılır. Bu durumla da daha çok stacklerde karşılaşılır.

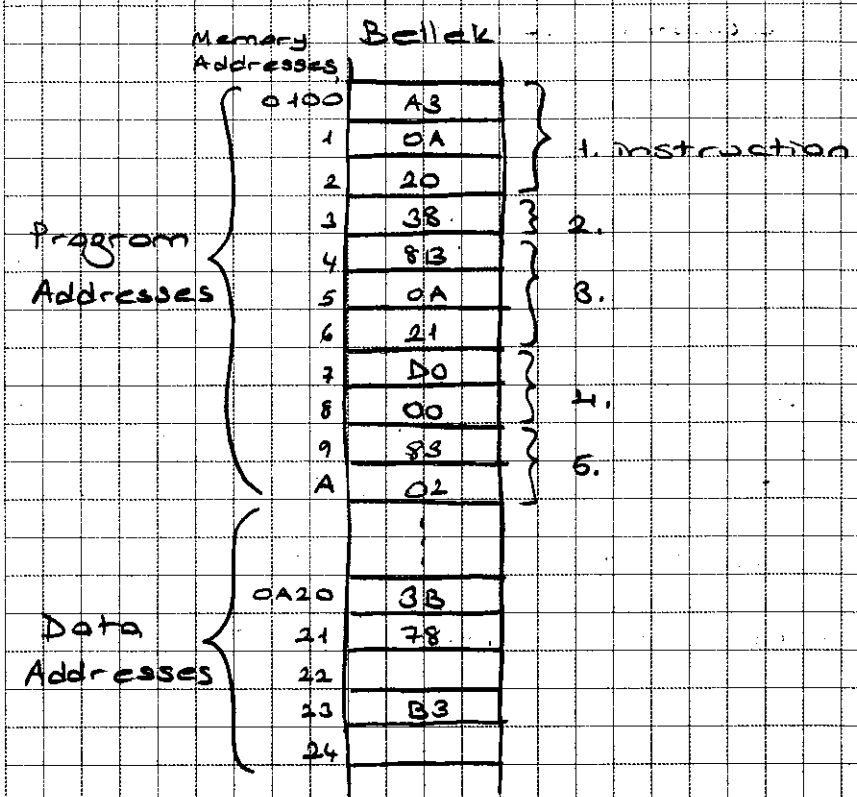
CPU Kaydedicinin Kullanılması:

Toplamanın adımları:

1- Toplanacak birinci sayının adresini belirle.
2- Bu adresteki bilgiyi akümülatöre taşı.
3- Toplanacak ikinci sayının adresini belirle.
4- Bu adresteki sayı ile 2. adımda akümülatöre taşınmış sayıyı topla.

5_ Toplamın bırakılacağı bellek adresini belirle.

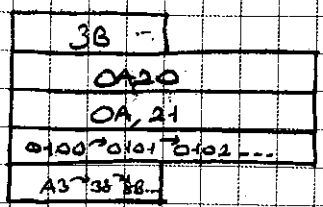
6_ Toplamı bu adrese yaz.



A3 kodlu bir emir oluşturuyorsa. Bu emirin görevi arkasından gelen 2 sayıyı data counter'a yüklemek olsun.

8B kodlu emir de arkasından gelen 2 sayıyı okur, index registere koyar

D0 kodlu emir de arkasından gelen değerle 3B'yi toplar. Sonucu akümülatöre yazar.



A (akümülatör)
DC (data counter)
IR (index register)
PC (program counter)
I (instruction)

- 0A20 → Base değer

00 gelince toplama yapılır. Toplam adres baştan çıkarılır. 0A20 yerinde kalır.

* OS, programın başlangıç adresini PC'ye koyar. Monitörlerde execute diye run'ın işlevini yapan bir emir vardır.

CPU'nun Reset i vardır. Bunun görevi registerleri sıfırlamaktır.

CPU PC'de bulunan değeri adres busla dışarı çıkarır. İlk adresteki değeri okur, kaşar, I'ya koyar.

- PC programın neresinde kaldığımızı gösterir.
- 38 → DC'deki sayıları dışarı çıkarır. Data bellekteki. İlk sayı A'ya gönderilir. PC'deki değer değişmez çünkü Data Mem.'ye ulaşıldı, Prog. Mem 'ye ulaşılmadı.

* 32 bitten itibaren cash kullanılır. Segment registerlar geliştirilerek cash bellek oluşturulmuştur.

- 80 → Kendinden sonraki değeri IR'ye ekler.

$$\begin{array}{r} 0A21 \\ + \quad 00 \\ \hline 0A21 \end{array}$$

offset (displacement) → 8 bitlikte

↓ PC'de

Sonra bu değerle akümülatördeki değeri toplar, sonucu akümülatöre yatar.

88 → Kendinden sonra gelen sayıyı alıp, IR'deki değerle toplar. Toplamın belirttiği adrese A'daki değeri yatar.

- END HALT JMP → 0108

JMP
01
0B

(en az 3 byte)

31	16	15	0	
		AH	AL	EAX
		BH	BL	EBX
		CH	CL	ECX
		DH	DL	EDX
		SI		ESI
		DI		EDI
		PI		EBP
		IP		
		SP		
		FLAGS		FLAGS

① ②

LDTR
GDTR

	15	0	63	52	51	20	19	0	
CS	FFFF		ACCESS		Base Address		Limit		CSDCR
SS									SSDCR
DS									DSDCR
ES									ESDCR
FS									FSDCR
GS									GSDCR

UniQue design

80486 CPU'sundan itibaren PC'lerdeki aritmetik co-processor (APU) CPU'nun içine eklenmiştir. Günümüzde data büyüklükleri: 128, 80, 64, 32 bit

① Grup: adreslemeye yarar

ESI = Source index
extern

EDI = Destination Index

EIP = Intruction pointer

ESP = Stack pointer

EBI = Base pointer (stack pointerin bitlerini tutarak nerde kaldığımızı gösterir)

CS = Code segment (kod bellek birimi)

SS = Stack "

DS = Data "

ES =

FS = File "

GS =

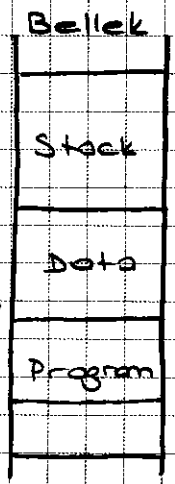
DCR =

LDTR = Local Data Table Register

GDTR = Global " " "

protected mode'da istenilen alanın dışına çıkılmak ve textlerin deforme olması engellenir.

Real mode: Bilgisayarın en ilkel çalışma şekli. Koruma yok, multi tasking yok.



CS = F000
IP = + 0400
F0400 → Fiziksel adres

Her taska erişimde güncellenen access kısmı sayesinde erişim sayısı ve en az kullanılan sayfa belirlenir. Böylece bu sayfa yerine yeni veriler yazılır.

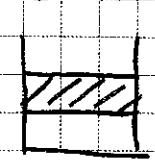
CS 16 bit
CS DCR 32 bit
} adres işinde kullanılan toplam bit sayısı 48 bit

CPU 32 bit

64 tera bayt (visual / sanal bellek)

2^{52} = 4 GB

Bir tablomuz var ve bu tabloyu adresleyen LDTR ve GDTR registerlerimiz var



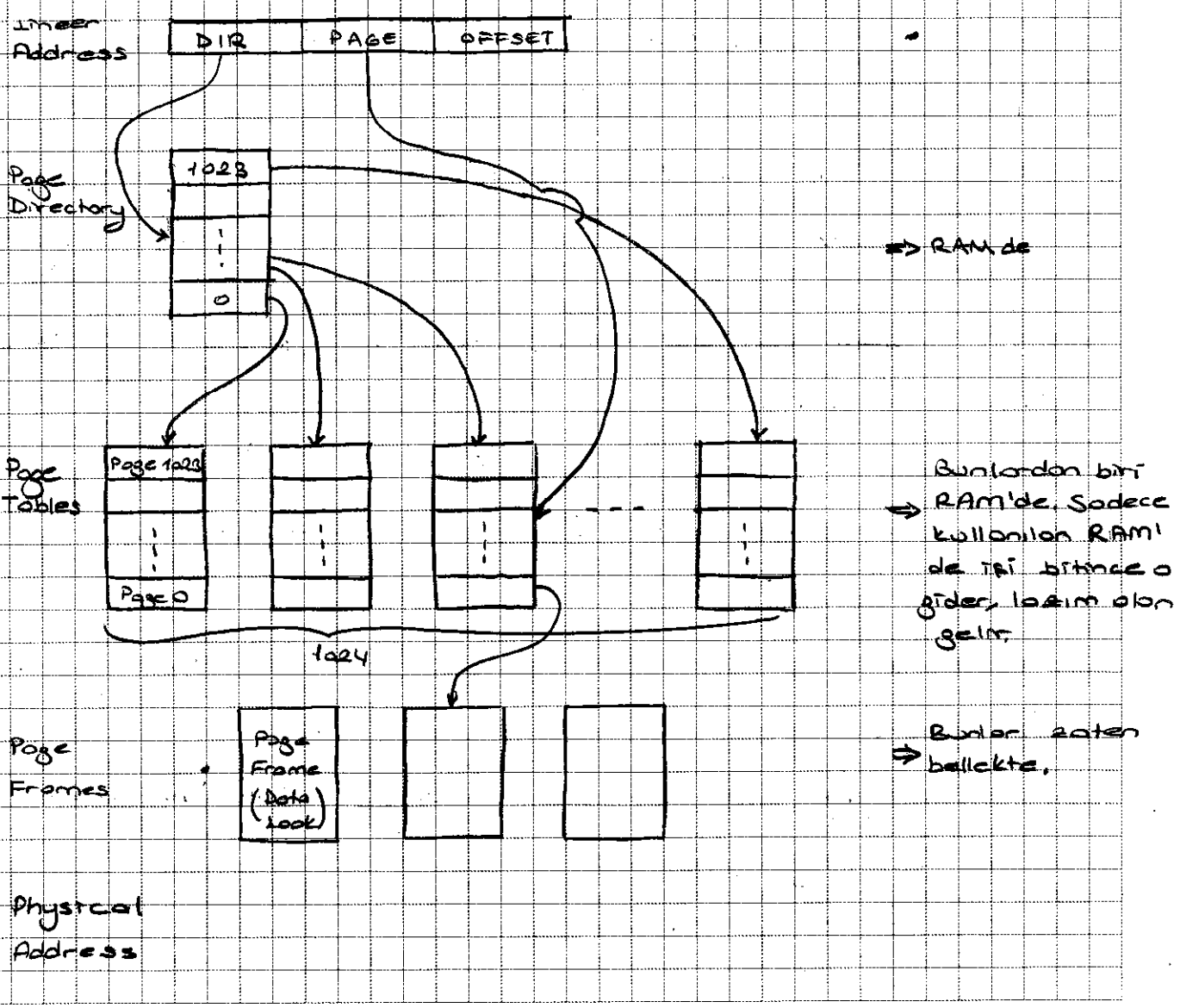
core memory

8086
86
286
386
486
pentium

} 16 bitlik (extern kısmı, FS ve GS yok)

} 32 bitlik

CS başı belirlenmiş bölgede nereye gidileceğini gösterir, Limit programların birbirini üstüne binmesini engeller, max. sınırı tutar,



2. Aritmetic Logik Birim (ALU)

Aritmetik, logik sistemleri yürüten bir birimdir.

a. Toplayıcı: 8 bitlik CPU'larda 8 bittir. Toplama, çıkarma, çarpma ve bölme işlemlerini bir tkt logik birimi de kullanarak toplayıcı ile yapabilirsiniz.

80386 CPU

80387 APU (coprocessor) → Aritmetik işlemleri yapmak için tasarlanmıştır. Belleği adresleme özelliği yoktur. Bu CPU (80386) ile birlikte kullanılır.

b. Boole işlemleri: Bitleri test etmek için gereklidir.

```
  011000
AND 010000
```

→ 6. bitin kaç olduğunu öğrenmek için o bite karşılık gelen yere bir koyarak test ederiz.

c. Tümleme: Çıkarma ve bölme işlemlerinde gerekli olan işlemdir.

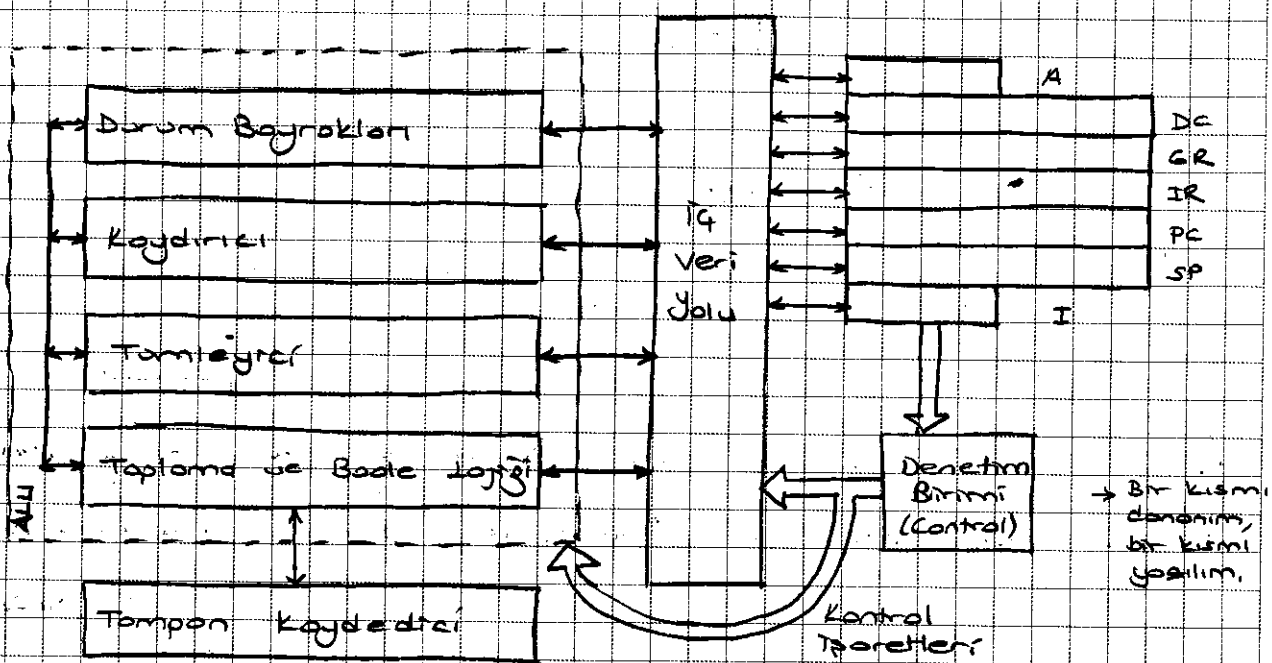
d. Kaydırıcı: Aritmetik işlemlerde ve bayroğa atmak için kaydırıcıya ihtiyaç vardır.

- Her CPU'da minimum bu dört birim bulunmalıdır.
- ALU'da bir de durum bayrakları bulunur.

3- Denetleme Birimi (CU)

Denetleme birimi genelde firmware düzeyde yapılır. CPU'nun elemanlarını yetkilendirir.

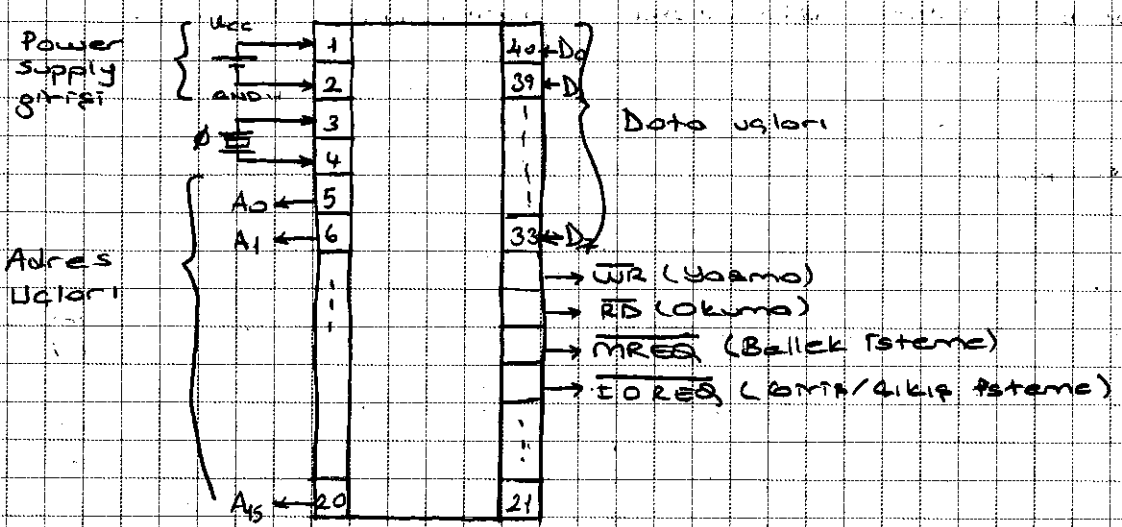
CPU'nun Yapısı:



Tampon kaydedici kullanıcıya açık değildir. Toplama işleminde CPU dışardan getirdiği sayıyı tampon kaydediciye kaydırak toplama birimindeki A'daki sayı ile toplar.

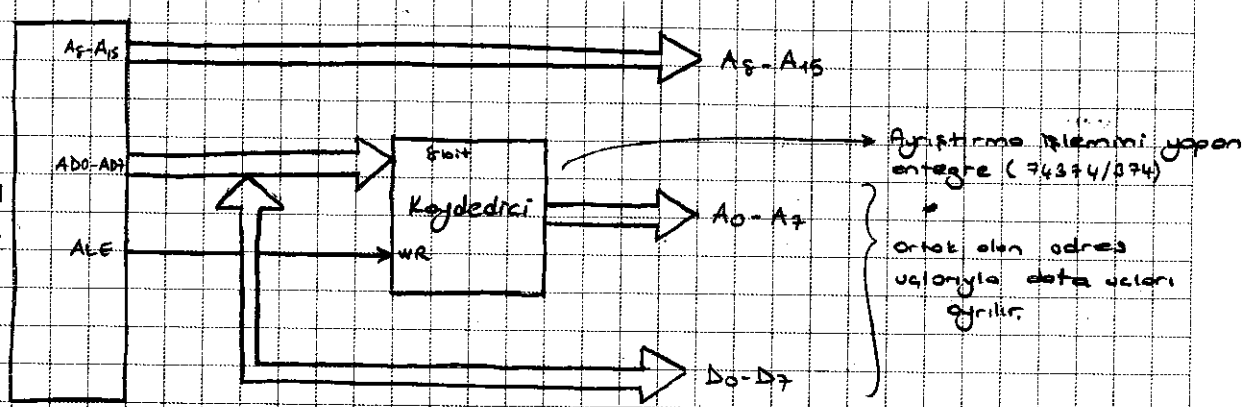
Kontrol birimi hem CPU içindeki hem de CPU dışındaki birimleri denetleyebilir. Bu işlemleri de mikro emirlerle yapar.

CPU'nun Bacakları (Pins)



CPU saatle kontrol edilen bir ardışıl devredir. CPU'nun en az 2 bacağı beslemeye ayrılmıştır. 8 bitlik işlemcilerde Vcc 5Volt, GND ise 0Volt'tur. Günümüz CPU'larında Vcc değeri 1.6 Volta kadar düşürülmüştür. CPU'nun 2 bacağı da saate ayrılmıştır. Yüksek frekanslı işler ya üretildiği yerde kullanılır ya da dışarıda üretilip taşının. Taşıma olayı yüksek frekanslarda çok zordur. Bu nedenle kaynak işaretin(düşük frekanslı) osilatör ve fuer dönüşümüyle CPU'nun içinde frekansı yükseltilir. Saati etkin kılmak için bir kristal ya da kondansatör takılır.

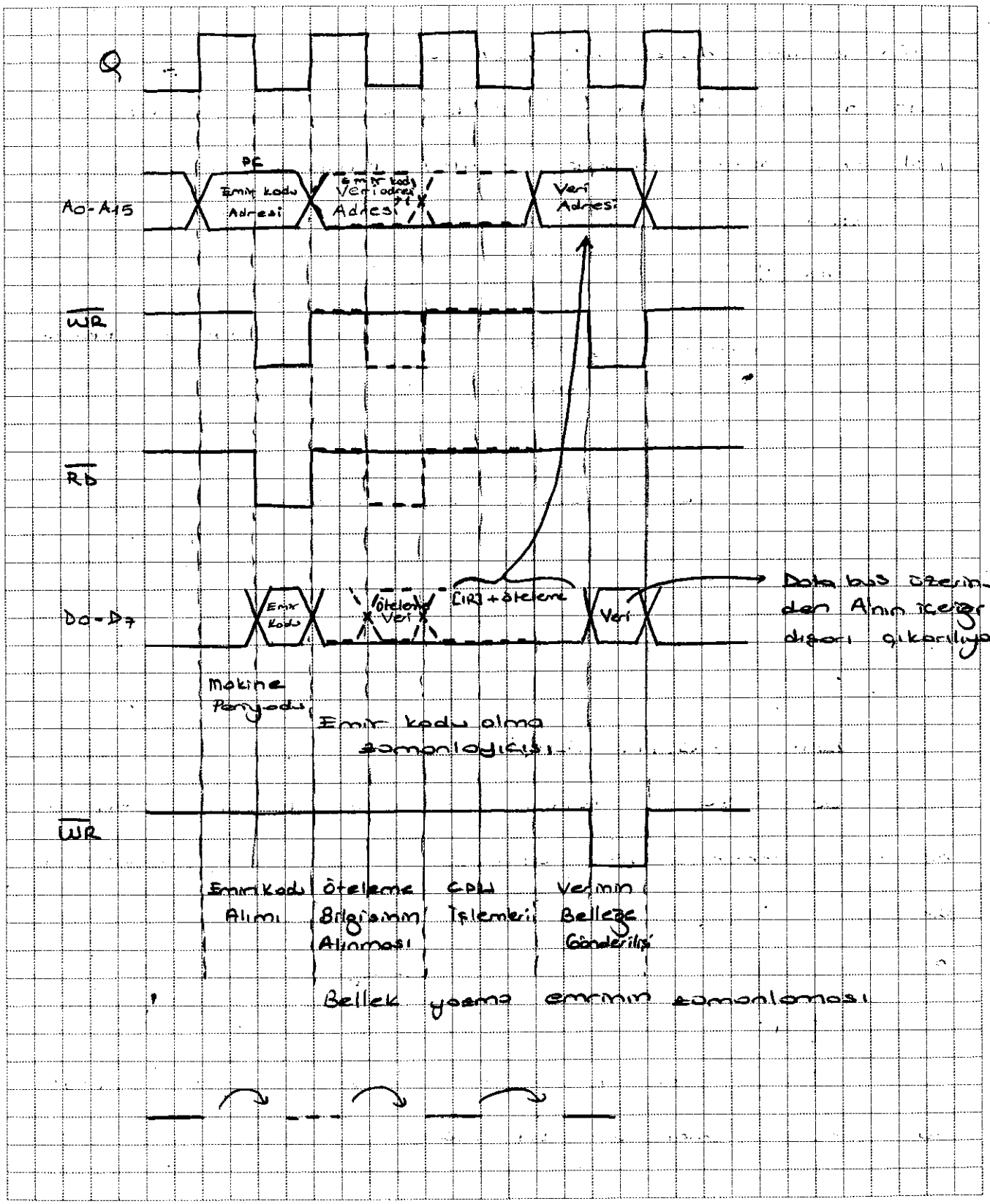
• Data buslar daha büyüktür. Örneğin CPU 32 bit, data bus 64 bit (1'den fazla data okuyabilmek için paralel çalışır). Modüler bellekte 8 bitlik modüller vardır.



• ALE (adres latch enable) sayesinde WR aktif olur. Adres uçlarının belirttiği bellek alanına data uçlarındaki bilgi yazılır.

• Aktif alçak: Bir işaret normalde yüksek gerilimde bekler, İş yapacağı zaman düşük gerilimde çalışır. Buna aktif alçak denir. Bunun sayesinde hem daha hızlı hem de daha az enerjiyle iş yapılır.

• Gömülü CPU'larında her bacağa ki göre yüklenmiştir. Ayrıştırma işlemi local bus'ta gerçekleşir. Global buslar tüm sistemlerde aynıdır, ama local buslar ise CPU üreticisine kalmıştır.


Q
PC
A0-A15
Emir kodu Adresi
Emir kodu / Veri adresi
Veri Adresi
WR
RD
D0-D7
Emir Kodu
ötelenip Veri
[IR] + ötelenir
Veri
Makine Periyodu
Emir kodu alma zamanlayıcısı
WR
Emir Kodu Alımı
ötelenip Bilginin Alınması
CPU İşlemleri
Verinin Belleğe Gönderilişi
Bellek yazma emrinin zamanlaması

- Makine periyodu = saat periyodu
- Her emir kodu alımı bellek okuma işlemidir.

İlk önce adres okunur (A_0 - A_{15}), daha sonra ortak hattan data gelir. Aynı hat hem adres bus hem de data bus olarak kullanılır.

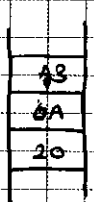
CPU ilk önce komutun adresini okur. Sonra $\overline{MREQ}$ belleğe erişim yapılacağını, $\overline{RD}$ ise bellekten okuma yapılacağını bildirir.

D_0-D_7 data yolları aktif yapılarak bellekten emir kodu alınır ve emir kaydediciye (I) yazılır.

$\overline{MREQ}$ her CPU'da yoktur. Genellikle Intel ailesinde bulunur.

Emir Karmaşıklığı

RISC ve CISC kullanılan bilgisayarlarda emirler hemen hemen aynı uzunlukta ve karmaşıklıkta olmalı.

- 

A3 data counteri 0A ve 20 ile yükler.

Bu karmaşık emri 2 basit emre ayıralım.

A4
0A
A5
20

A4 deta sözcüğüm anlamlı kısmını 0A ile yükler,
A5 " " anlamsız " 20 "
yükler.

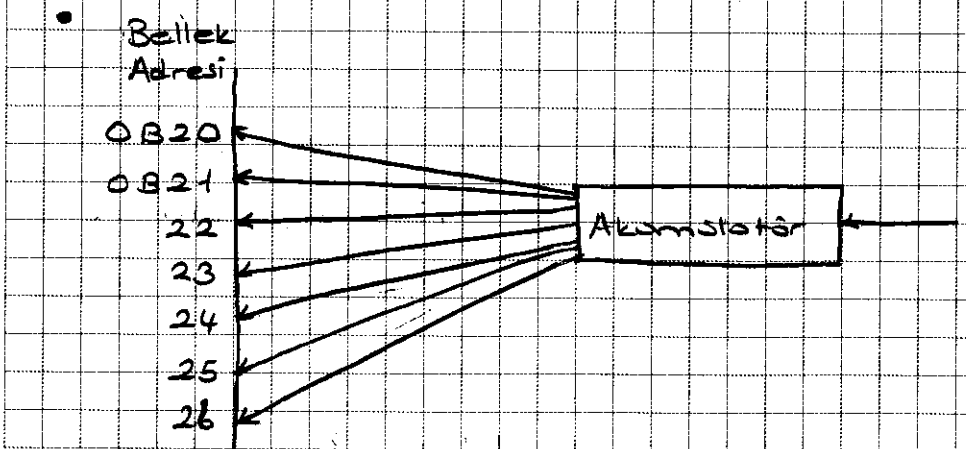
İcra süresi uzar ve bellekte kapladığı yer artar.

39
0A
20

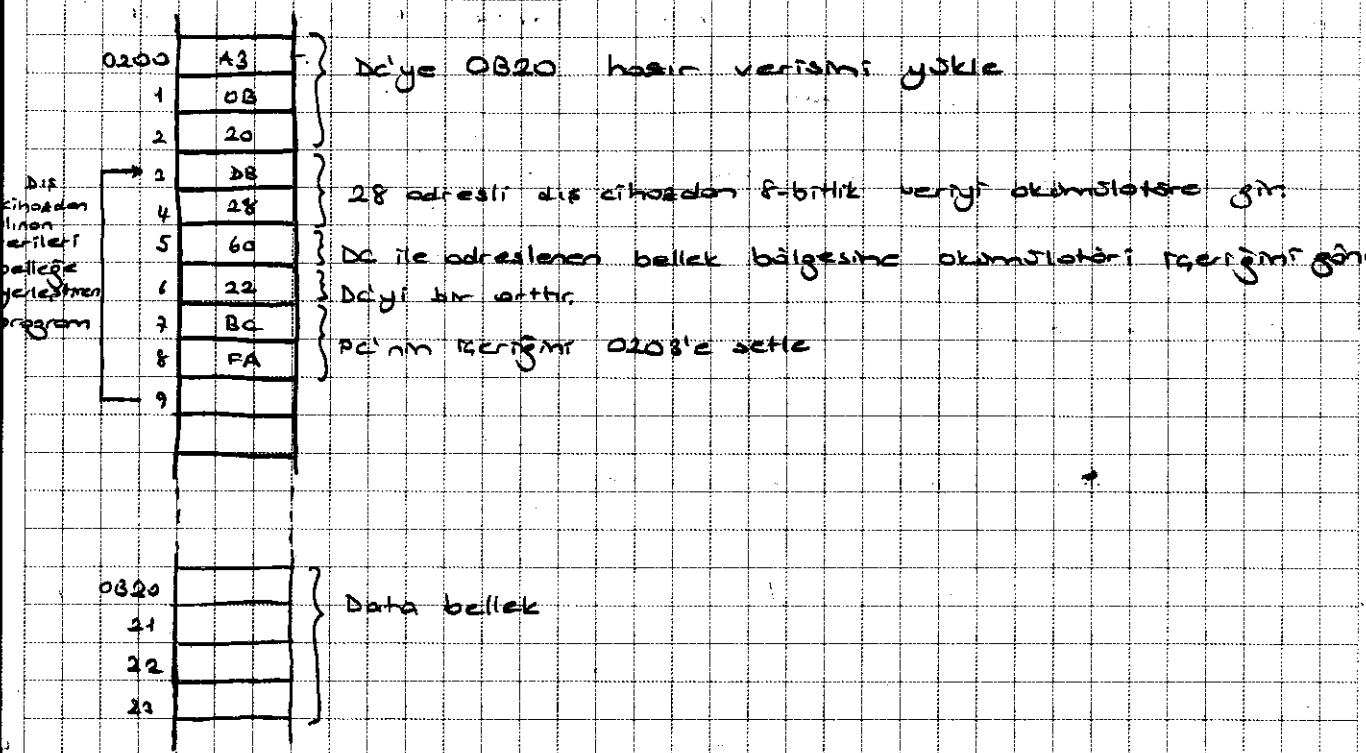
39 emri 0A20'yi alıp adres busa koyar ve arkasından 0A20'deki bilgiyi alıp A'ya koyar (doğrudan adres)

İcra süresi 4 saat periyodu uzar.

* Eğer kullanılan bir bellek alanı (mesela 0A20) küçük güncellemelerle tekrar tekrar kullanılacaksa DC'ye aktarılması, kullanılmayacaksa direk adresleme yapılması avantajlıdır.



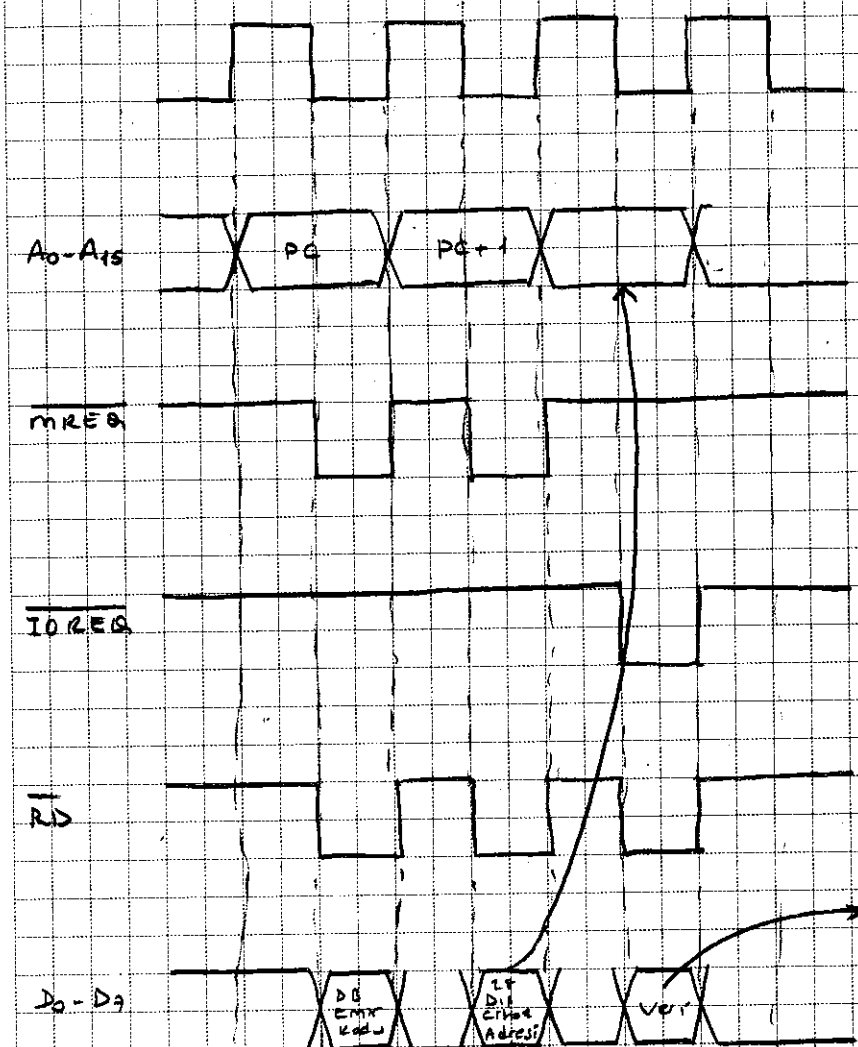
Veri tablosunun referanslanması.




```
IN   A, port
OUT  port, A
LDA  memory, A
LDA  DC, A
```

- Bu programda IOREQ kontrol hattı oluşturulacak.
- DB emir kodu 28 dış cihaz (port) adresi, DB 28 hattı adresten alır, A'ya yazar.
- BD (branch offset) : Döngünün sonuna dönerek kalmasını sağlar. Loop sayesinde döngünün başına döner.
- Emir kodunun belleğe mi dış cihaza mı ait olduğunu

anlatmak için IOREQ hattı kullanılır (dış cihaz için)



Giriş emrinin tamamlanması.

Q209

\+ FA

0209

\+ FFFA

1←0203

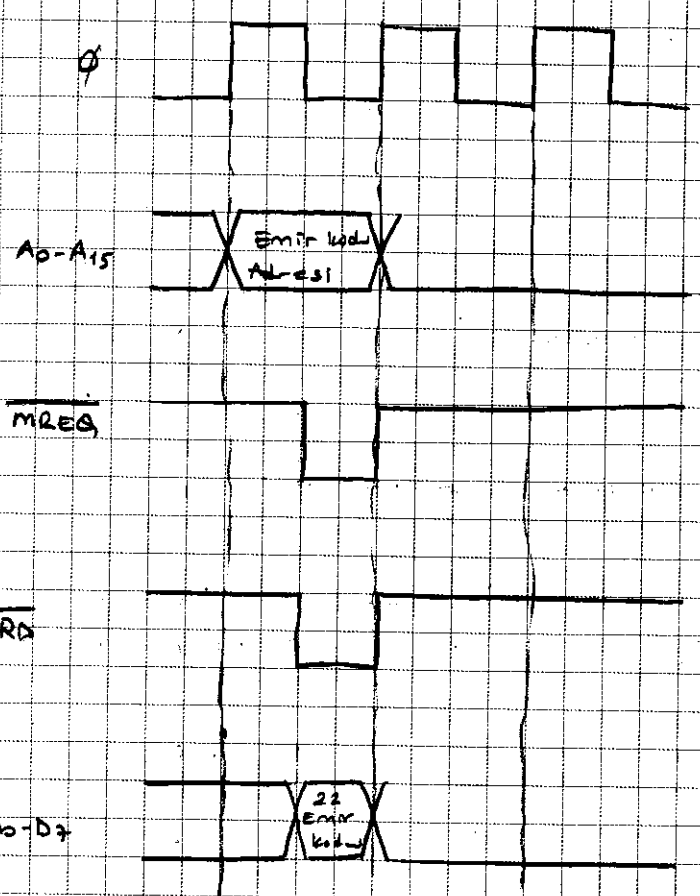
Bit sayısı eşit olmalı (+, -, *). Bilgisayar eşitler.

/ için 2 bitlik sayı 4 bitlik sayıya bö-

lünür.

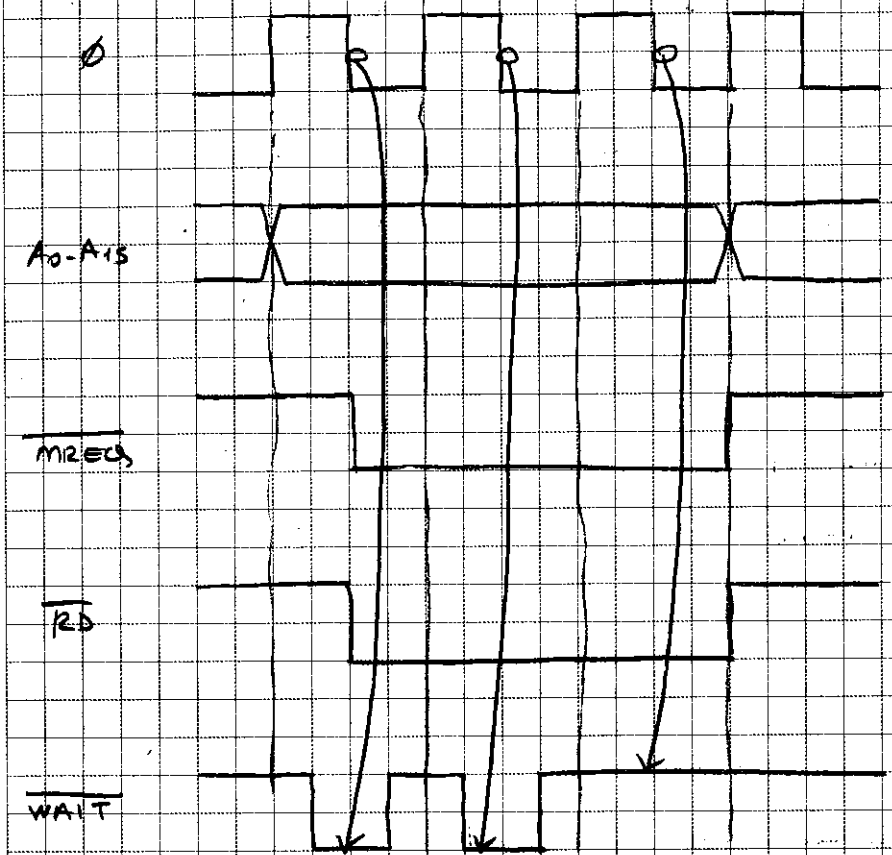
JMP adres → Bunu kullanacak mutlak adresleme yapmış olursa, Programın yeri değiştiğinde JMP'nin gideceği adres yanlış olacağından program çalışmayacaktır. Güncelleme gerektirir.

Bağıl olarak → Program bağıl adresle dönüşür, yani bize belli bir adres yerine kaç bit ileri veya geri gidileceğini söyler. Programın yeri değiştiğinde de sorun çıkmaz.

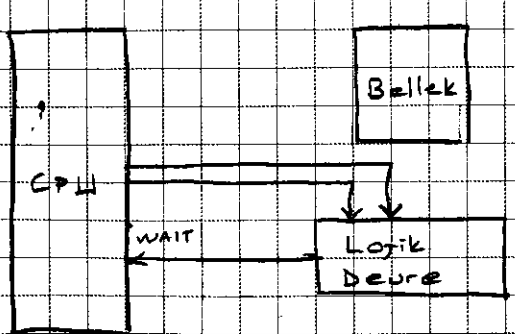


CPU işlem emrini tamamlaması

8086'dan sonraki CPU'lar modüler bus zamanları da değerlendiriyor.



Bekleme durumu zamanlaması



Ara bellek CPU'nun hızına erişemez. Bu sebeple bu 2 cihazın hızını bilen Lojik devre WAIT sinyali üretip CPU'yu yavaşlatıyor.

- CPU WAIT işaretine bakar istek varsa bir süre yavaşlar. Tekrar bakar hala istek varsa daha da yavaşlar. Bir daha bakar istek yoksa bir daha bakmayı keser çünkü yeterli hıza düşmüştür.
- Ya cash vardır çünkü bellek yavaştır ya da bellek hızlıdır O halde çok pahallıdır ve cashe ihtiyaç yoktur.
- Minimum donanımla her işlemi yapabilen CPU'da

Toplama için → toplayıcı

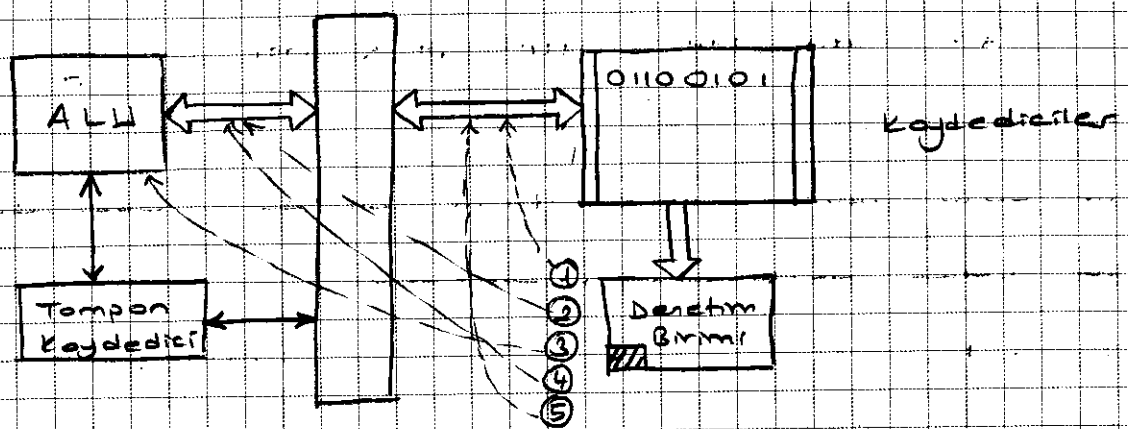
Çıkarma için → tümleyici

Çarpma ve bölme için → kaydırıcı ve toplayıcı

olmalıdır. Bu CPU daha çok software ağırlıklı çalışır. Günümüzdeki CPU'larda her çeşit sayı için ve her çeşit işlem için ayrı donanımı bulunur.

- Dünyada tek chipe dayalı ve ayrık olmak üzere iki çeşit CPU vardır.
- Her emrin ilk makine periyodu emir kodu alınmıdır.

INCA



① Denetim birimi okunmalıdır için iç veri yoluna taşır.

② İç veri yolundaki veri ALU'ya aktarılır.

③ ALU'daki değer 1 arttırılır.

④ Arttırılan değer ALU'dan iç veri yoluna taşınır.

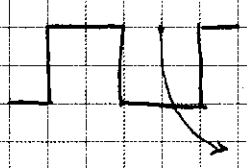
⑤ İç veri yolundaki değer okunmalıdır'e aktarılır.

- CPU'nun kaç bitlik olduğuyla mikro emirlerin kaç bit olduğu arasında bir ilişki yoktur. Örneğin 8 bitlik CPU'lardan 9 bitlik mikro emirler olabilir.
- Mikro emirlerin bit uzunluğu bizim CPU'muz için 9 bit seçilmiştir. Dolayısıyla 512 farklı seçme işareti üretilebilir. Bunun sonucu olarak CPU içinde ve dışında 512 farklı birim kontrol edilebilir.

Emir Kodu Alımı Mikroprogramı

Emir Numarası	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	Görevi
1	1	1	1	1	0	1	1	0	1	PC'yi AR(adres register)e taşır. PC'nin kopisi açılıyor ve AR enable yapılarak PC'nin işaretli veri AR'ye aktarılıyor. Böylece emrin adresi Adres Bus üzerinden alıcıya veriliyor.
2	0	1	1	0	0	1	1	1	1	RD ve MREQ'i aktif, WR ve IOREQ'i yüksek yap.
3	1	1	1	0	0	1	0	0	1	PC'nin anlamsız yorumu iç veri yoluna taşı. 8 bitlik işlemcilerde gönderilme ALU'da yapılır. 16 ve 32 bitliklerde ayrı bir işlem birimi vardır. PC'nin içeriği güncellendi.
4	1	1	1	1	0	0	1	1	0	İç veri yolunu ALU latchlarına taşı. PC'nin anlamsız yorumu geldi.
5	0	1	1	1	0	0	0	0	0	ALU latch içeriğini 1 artır.
6	1	1	1	1	0	0	1	0	1	ALU latch içeriğini iç veri yoluna taşı
7	1	1	1	0	0	1	0	1	0	İç veri yolunu ALU latchlarına taşı

8	1	1	1	1	1	0	0	0	1	PC'nin anlamlı baytını iç veri yoluna taşı
9	1	1	1	1	0	0	1	1	0	İç veri yolunu ALU latchlerine taşı
10	1	0	0	0	0	1	0	1	1	Elde(C)=0 ise bir sonraki emri atla.
11	0	1	1	1	0	0	0	0	0	ALU latch içeriğini 1 artır.
12	1	1	1	1	0	0	1	0	1	ALU latch içeriğini iç veri yoluna taşı
13	1	1	1	1	1	0	0	1	0	İç veri yolunu PC'nin anlamlı baytına taşı
14	1	1	1	1	1	0	1	0	1	Veri Registre (UR)'yi iç veri yoluna taşı
15	1	1	1	1	0	1	0	1	0	İç veri yolunu IL'ye taşı

-  13. emirden sonra dış cihazın veriyi gönderip göndermediğine bakılır (yani WAIT'in yüksek olup olmadığına bakılır yüksekse) veri gönderilmişse 14. adıma geçer.
- CPU'daki bir makine periyodu en uzun emrin adımına bölünür Diğer emirler de buna göre işletilir örneğin en uzun emir 25 adım, 1 ma-

kine periyodu 25'e bölünür.

- CPU dışarıya gelecek emir (örneğin bellekten) uzun sürelidir, CPU bu 15. emir kodunun dışarıdan gelecek cevabı beklerken yaptığı için zamandan bir kaybımız yoktur.
- Faz farkı olan saat darbeleriyle saat darbesi alt kısımlarla sağlanır.

CPU lerde makina periyotları tek saat periyodundan oluşmaz. Zaman kaybını engellemek için bir makina periyodu çok saat periyodu kullanılır ve böylece boş dilemler diğer saat periyotlarında kullanılır.

BELLEK (MEMORY)

Yarı İletken Bellekler

1. Yazılıp Okunabilen Bellek, RAM (Random Access M)
2. Yalnız-oku Bellek, ROM (Read-Only Memory)
3. Programlanabilen ROM, PROM
4. Silinip-programlanabilen ROM, EPROM/E²PROM, Flash memory

RAM

(1) Statik RAM

a) Statik bipolar RAM — I²L RAM

b) Mos RAM

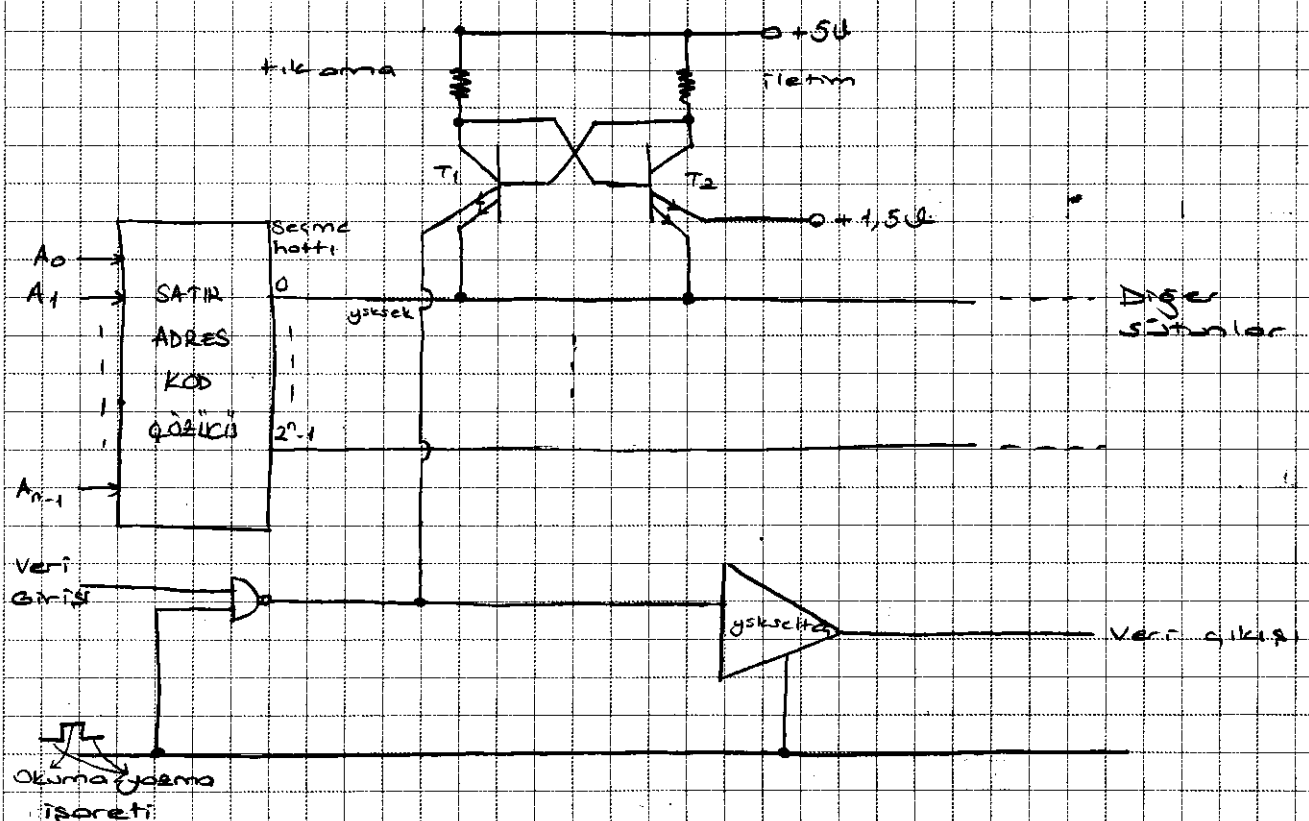
(2) Dinamik RAM

Statik RAM'lar hızlıdırlar (özellikle statik bipolar), fazla güç çekerler, pahalıdırlar. Gerilim var oldu-ğu sürece içeriği kaybolmaz. Ama bellek yapımına elverişli değildir.

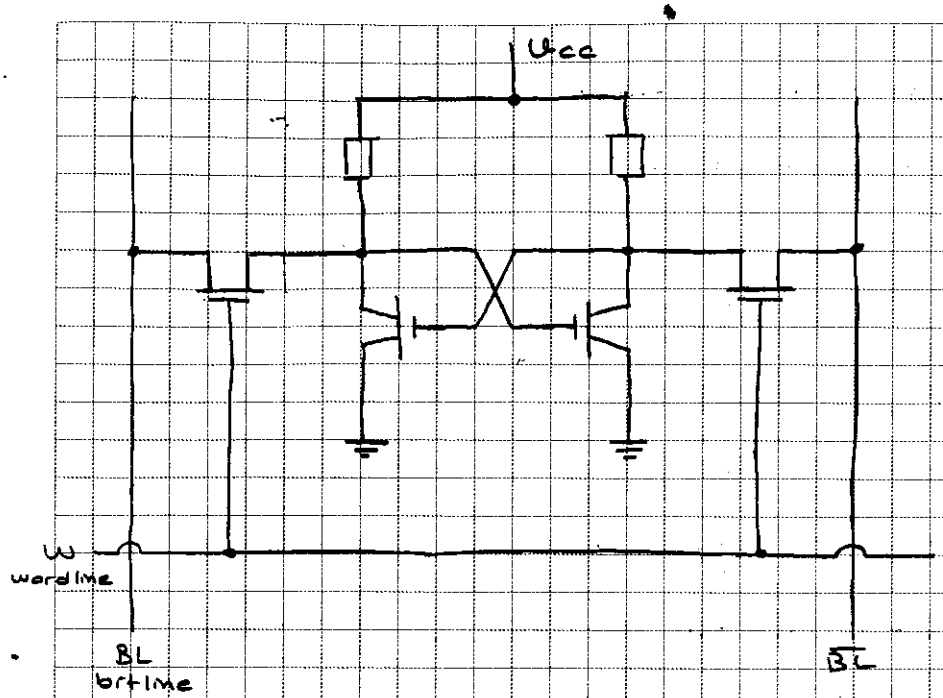
Dinamik RAM'lar az güç harcarlar, kapasiteleri fazla, fiyatları düşük, hızları ise statiğe göre azdır. En büyük dezavantajı gerilim kesilmediği halde içeriği 10 ms sonra kaybolur. İçindeki bilgi-lerin kalıcılığı için refreshleme gerekmektedir.

- Cache bellek statik, ana bellek dinamik yapılıdır.
- Dinamik RAMler maslardan yapılır.

Bistable multivibratör (iki kararlı devre)

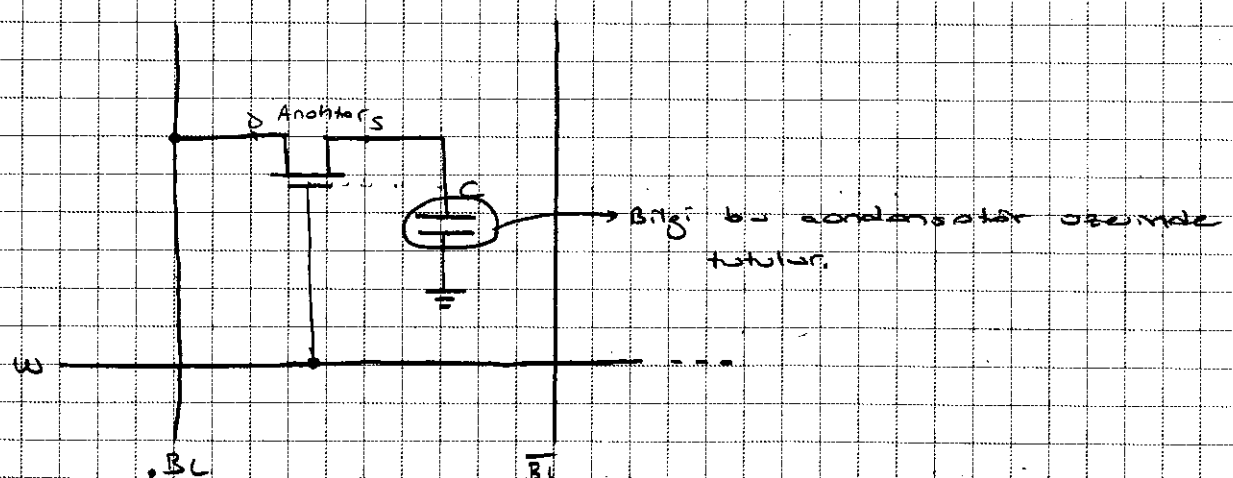


- Adrese bağlı olarak seçme hatlarından biri yüksek olur.
- Statik bipolar RAM'de bütün bellekler in adres girişleri vardır. Dolayısıyla her bellekte adres kod çözücü yo da adres kod çözücüler bulunur.



SRAM Memory Cell

2- Dinamik RAM :



DRAM memory cell

W'den gelen akım sayesinde anahtar kısa devre o-
du ve C'ye bilgi geldi. yani bilgi bu kondansatörde

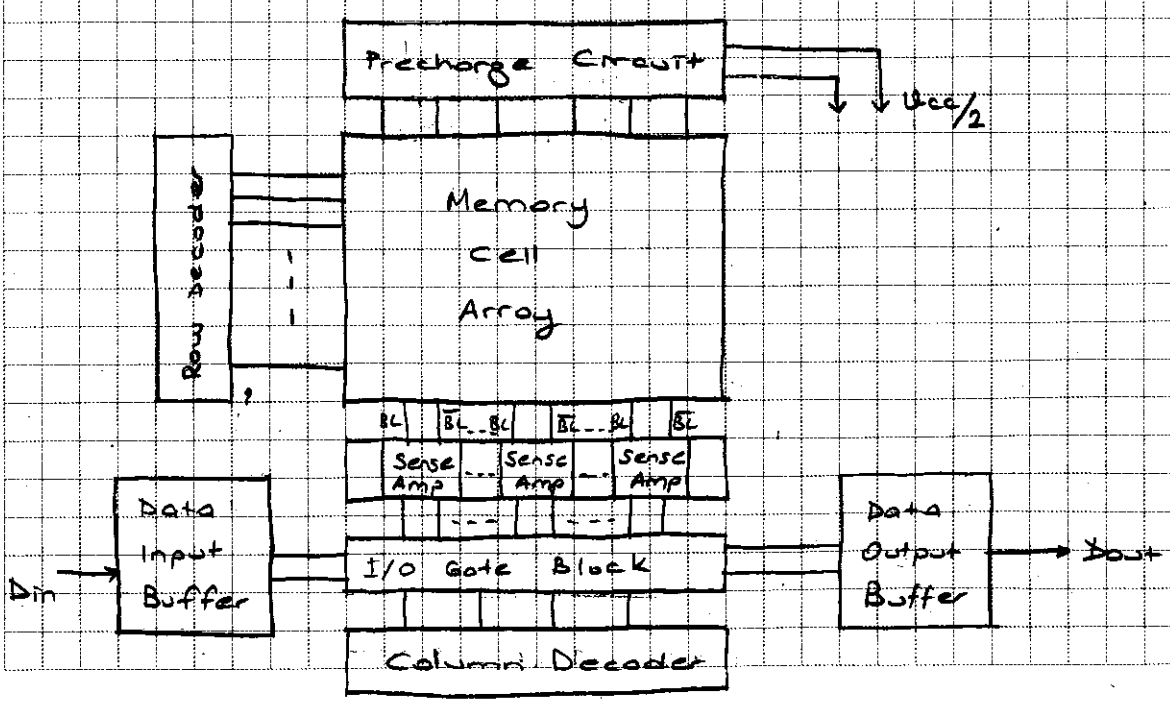
yük olarak tutulacak.

Statiklere göre çok daha fazla bilgi tutabilir.

Transistör sayısı daha.

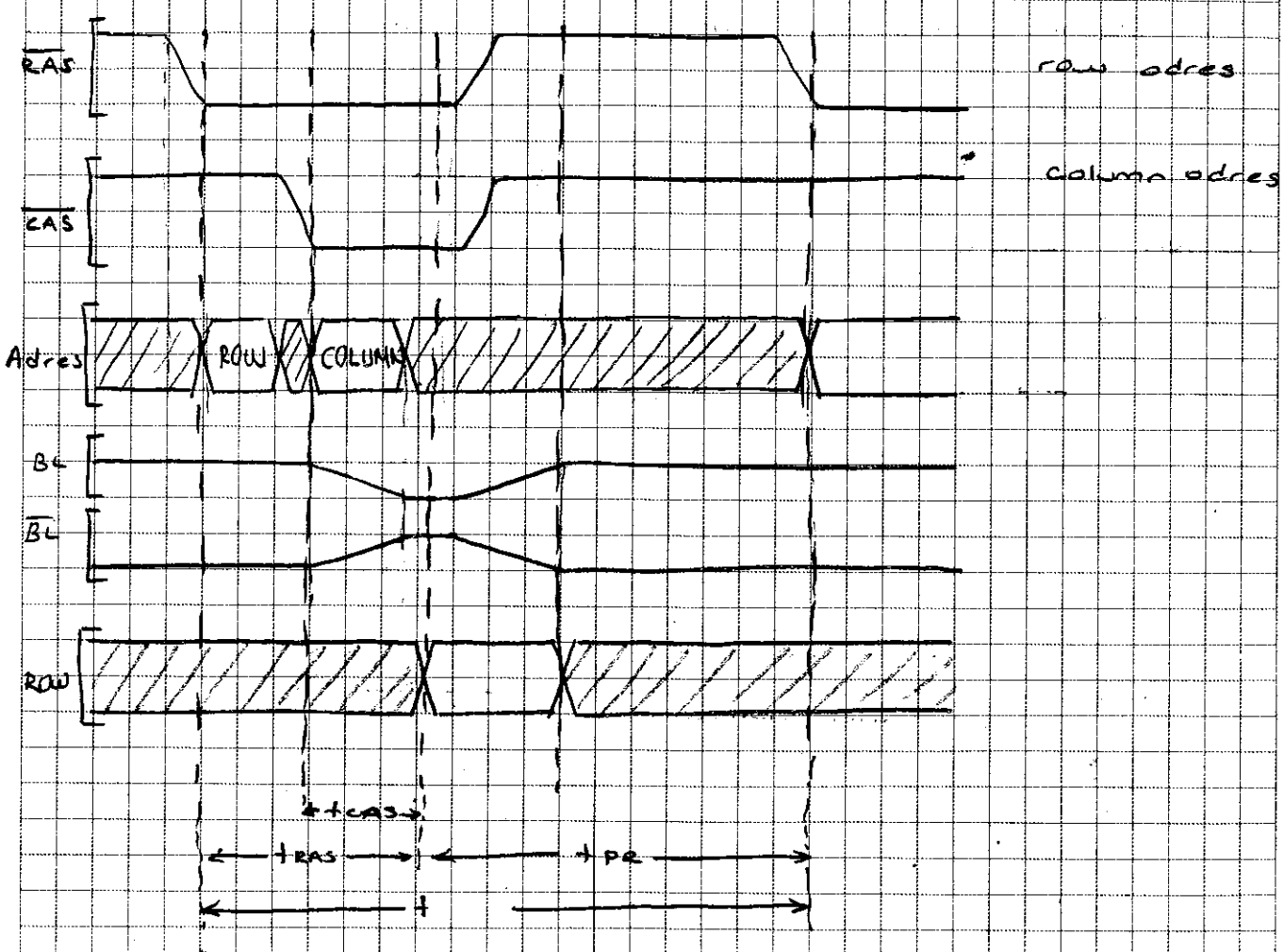
Dinamiklerin dezavantajlarından biri yaklaşık 1-10ms'de bir bilgileri yenileme (refreshleme) gerçekleşmesidir. Aksi takdirde bilgiler silinir. Bu refreshleme mekanizması (DMA) yakın zamana kadar belleğin dışındaydı. Yani bu dışarıdan müdahale hızı düşürüyordu. Günümüzde bu mekanizma belleğin içine alınmıştır.

Bu kondansatör 100mV'luk gerilim demektir. Başladığında bunun altına düşeceğinden veriye değerliğini kaybedeceğinden çok fazla boşalmasına izin vermeyiz.



UniQue design

Adres bacaklarının sayısı azdır. Çünkü bir bilginin önce bir yarısını chipin içine yolluyor, sonra diğer yarısını... Bu hızı düşürmez çünkü yarısı gidiyor mesela ikinci diğer yarısı da gider.

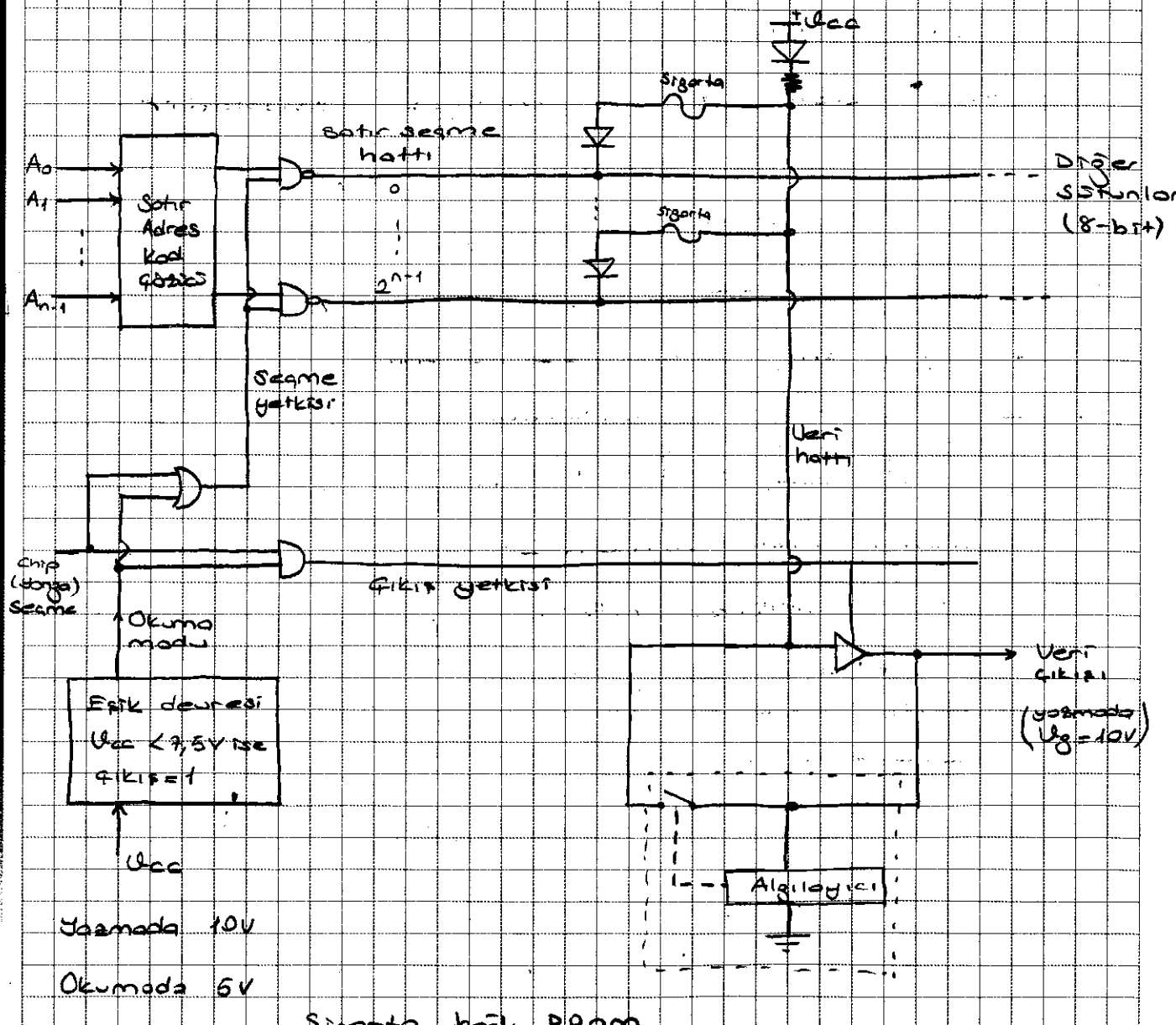


→ KALAN KISMA KİTAPTAN BAKIP, SORUMLUSUN ! ←

tPR → Bir işlemden sonra yeni işleme geçerken önceki işlemin etkisini ortadan kaldırmak için ihtiyaç duyulan süre. (Yavaş olmasının sebebi!!!)

Programlanabilen ROM (PROM)

ROM'da kullanılmayacak hücrenin transistörleri üretilmez. Ama PROM'da hangi hücrenin kullanılacağı önceden bilinmediğinden bütün hücrelerin transistörleri üretilir. Bu sebeple PROM ROM'a göre daha pahalıdır.



Yazmada 10V

Okumada 5V

Sigorta bağlı PROM

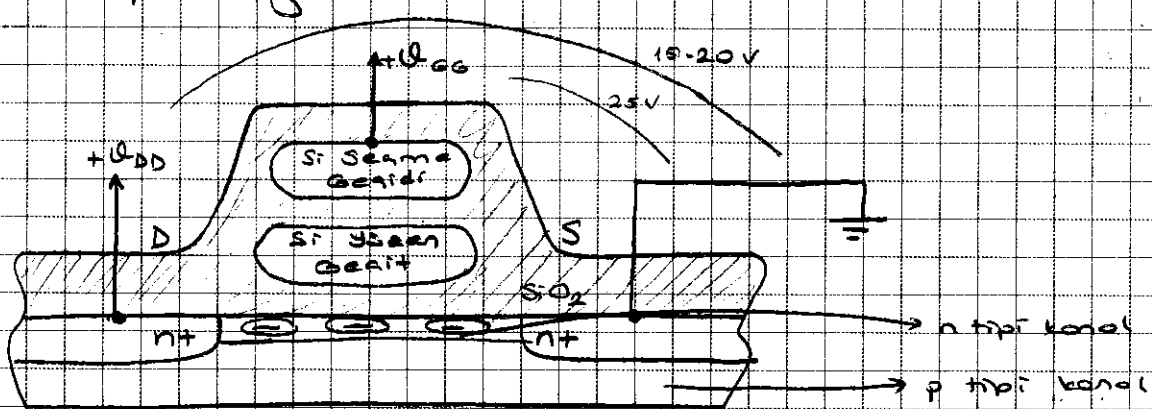
Yazma işleminde yeterli bilgi akış yönünden PROM'a gelir.

$W = U \cdot I \cdot t = \frac{U^2}{R} \cdot t$ ... sigortayı yakan enerjidir.

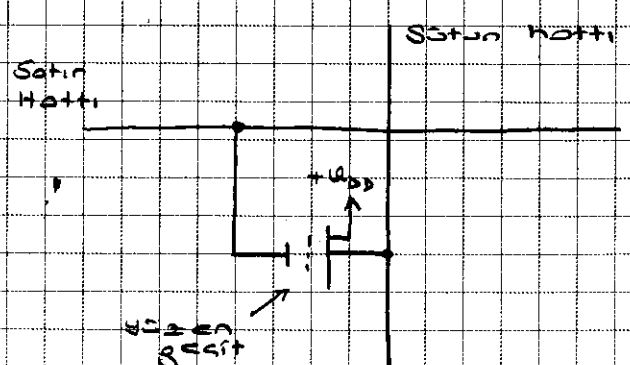
↓ enerji

PROM bir kereye mahsus bizim tarafımızdan programlanır, daha sonra yalnız okunabilir.

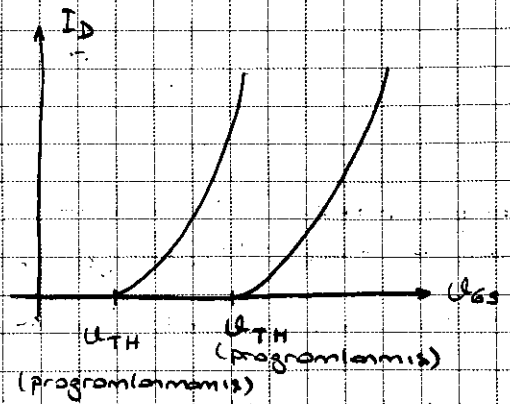
Silinip-Programlanabilen ROM (EPROM)



EPROM hücresinin kesiti



EPROM hücresinin sembolik gösterimi



→ Geçit yükünün EPROM MOSFET iletkenliğine etkisi

EPROM → Elektrikle programlanır, optik olarak silinir,

E^2PROM → Elektrikle programlanıp silinir,

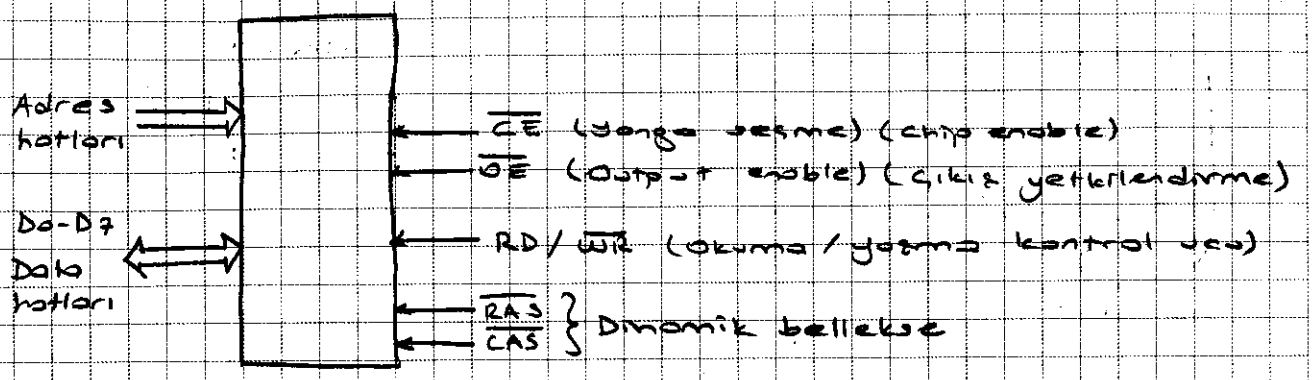
Yükler Si-yüzey-geçit ortamında toplanır. GS arasında akım akmaz, yani sigorta atmış gibi olur.

Si-yüzey-geçit ortamındaki elektronları ışık veya elektrik yardımıyla yerlerine gönderirsek GS arasından akım akar, yani belleği silmiş oluruz.

Flash bellekler de bir E^2PROM'dur. Farkı 1 hamlede silinebilmesidir. CPU'dan gelen komutla birden silinir.

EPROMlar E^2PROMlardan daha hızlıdır çünkü E^2PROM'da daha çok transistör bulunur.

Belleğin uçları

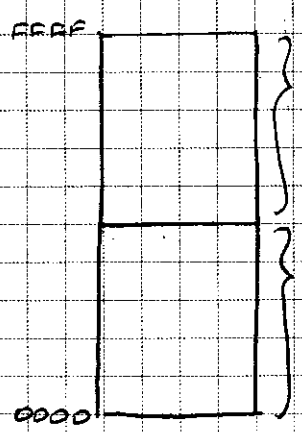


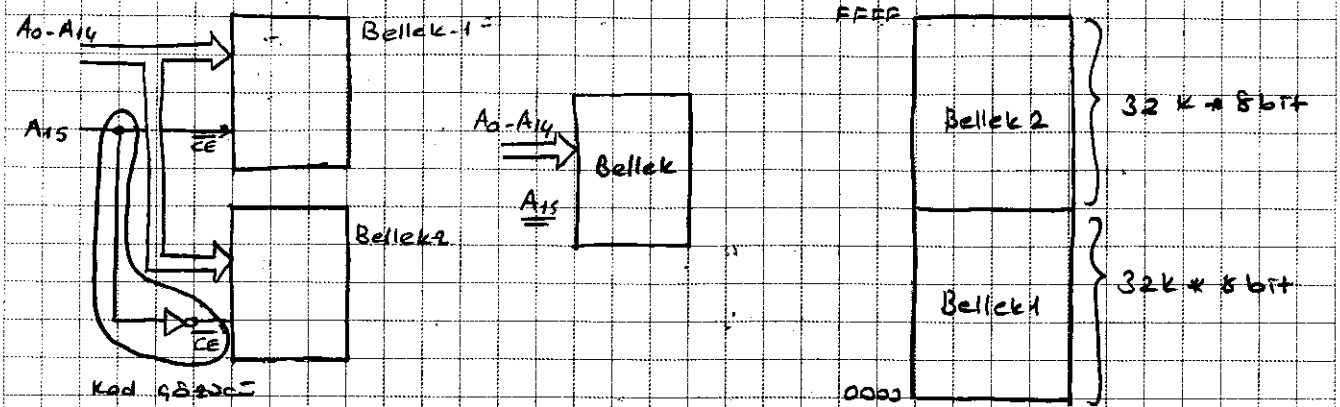
Chip seçilmemişse o belleğe ne yazabiliriz, ne de okuyabiliriz.

OE, çıkış yetkilendirilmemişse o bellekten çıkış alınamaz. Ancak belleğe bilgi yazabiliriz.

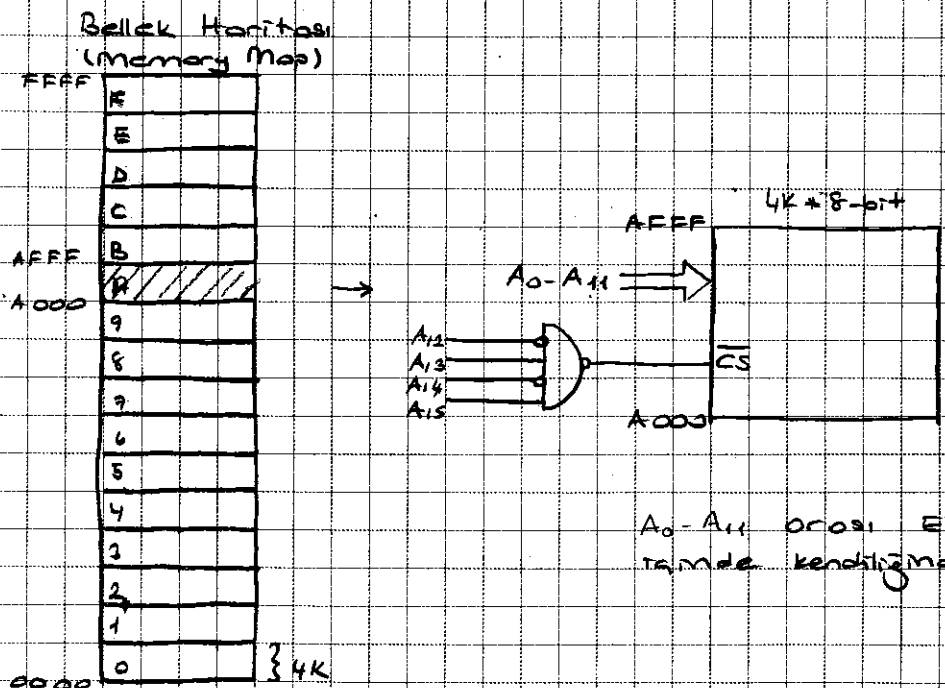
Bellek Uygulamaları

Kod çözücüler (demultiplexer = decoder)





ÖR! 4K * 8-bitlik bir belleğimiz olsa ve biz bunu A000 adresinden itibaren yerleştirmek istesek kod çözücü miz ne olmalı?



A0-A11 arası EPROM'un tamamı kendiliğinden gidiyor.

$\frac{64K}{4K}$ = 16 tane 4K'lık olan

2^{12} (12 = adres uç sayısı) tane and kapısı vardır.

A 0 0 0

A F F F

↓

12 and kapısı seçeneğe

bunun kodu çözülür.

Biz Alacın kodunu çözeceğiz.

A = 1010

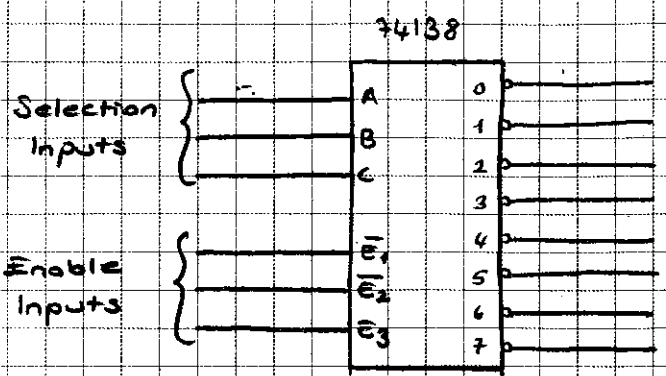
Doğruluk tablosunu yaparsak.

A_{15}	A_{14}	A_{13}	A_{12}	F_0	F_1	F_2	F_3	---	F_{16}
0	0	0	0	1					
0	0	0	1	1					
1	0	1	0	0					
				1					
				1					

$f_0 = \overline{A_{15}\,\overline{A_{14}}\,A_{13}\,\overline{A_{12}}}$

74138
74132
} Hazır bulunan basit kod çözücüler. Bunları

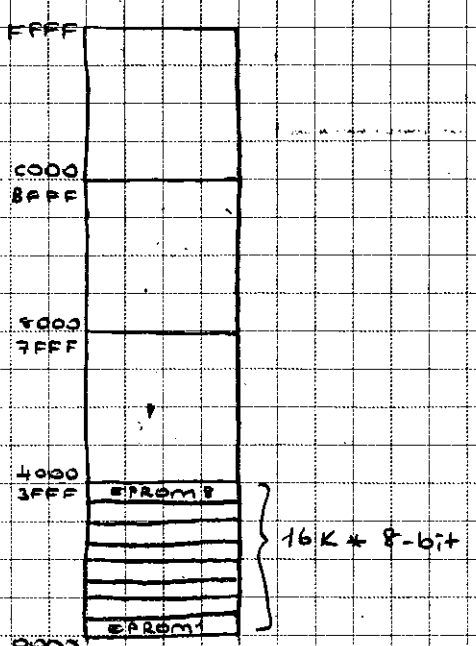
kullanırsak Tasmiz daha da kolaylaşır.



C	B	A	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{7}$
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Bu chipte 8 tane and kapısı var ve dolayısıyla 8 çıkış var. Bu çıkışlar bize veriliyor. Biz and-or logiği kullanmaktan kurtarıyor.

Ör: 2716 EPROM 2K * 8-bit



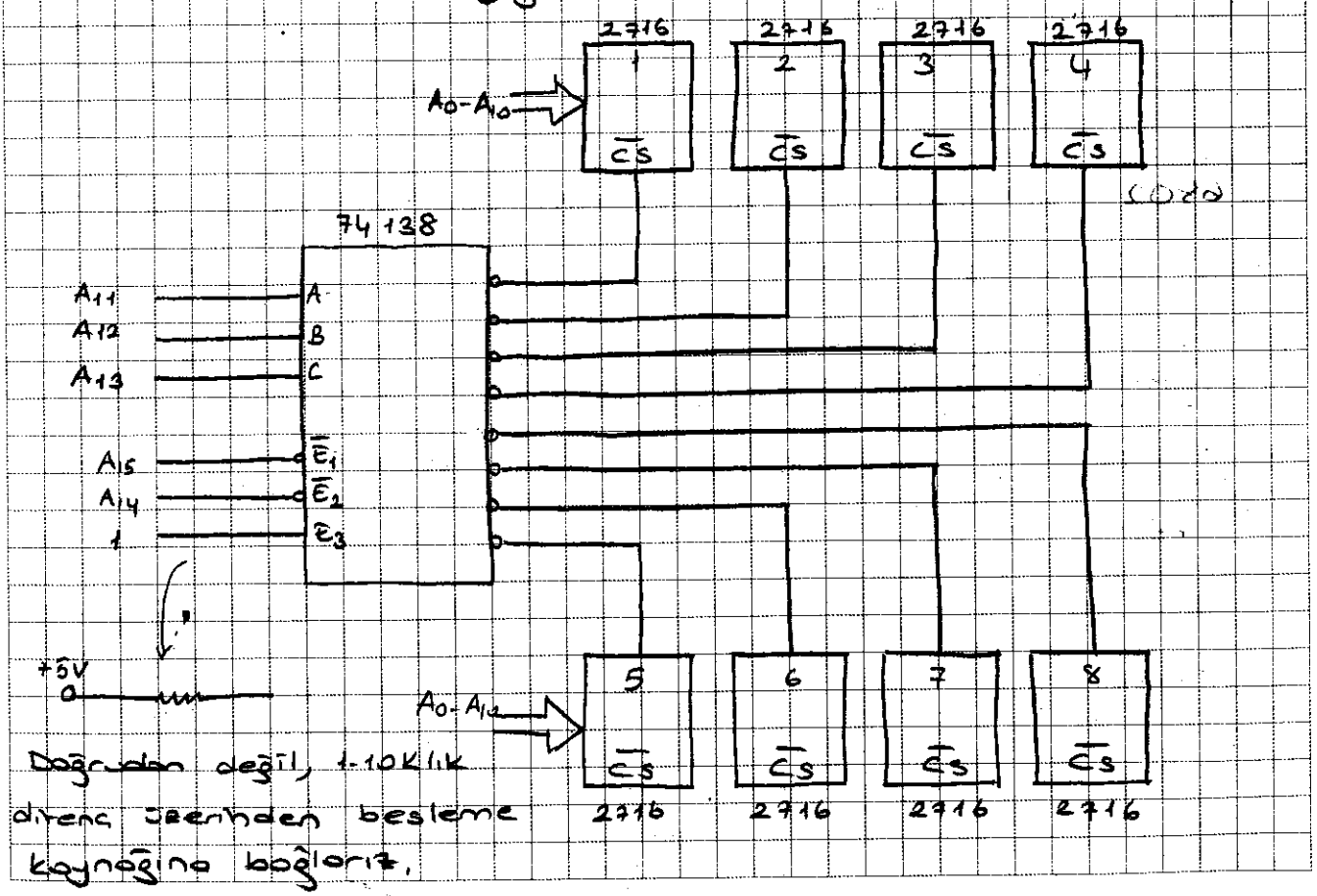
UniQue design

$\bar{E}_1$	$\bar{E}_2$	C	B	A					
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	---	A_0	
0	0	0	0	0	x	x	---	x	0000
0	0	1	1	1	x	x	---	x	3FFF

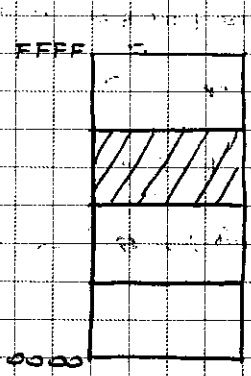
değişim yok — 0-3 arası değişim — chipin tamda gözüktüğü

Eğ extra. Dışardan lojik-1 uygularız.

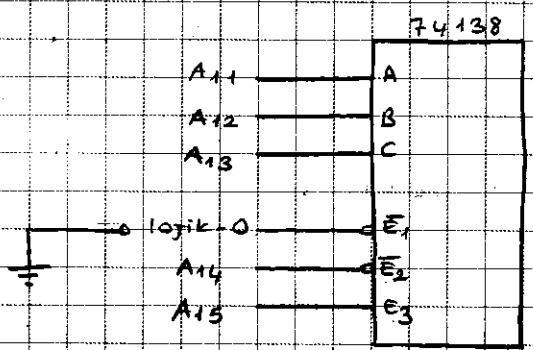
Değişmeyen bitleri Enable ucuna, değişken bitleri selection ucuna uygularız.



Doğrudan değil, 1-10K'lık direnç üzerinden besleme kaynağına bağlarız.



A_{15}	A_{14}	
1	1	
1	0	→ Bunu seçmek istesseydik,
0	1	
0	0	



6802 CPU'sunun Emir Takımı (Software)

Çok fazla emir yok. Emir kodları 1 bytetır.

Mode	Execution Time (Cycle)	Instruction Code (Hex)	Source Listing Example	Explanation
A Immediate	2	89	ADC A #$25	Note 1
A Direct	3	99	ADC A $25	Note 2
A Extended	4	B9	ADC A $7168	Note 3
A Indexed	5	A9	ADC A $25,X	Note 4
B		C9		
⋮		D9		
B		F9		
		E9		

Note 1: (C) + (A accumulator) + hex 25 → Result is placed in A accumula

Note 2: " " (hex address 25) → "

Note 3: " " (hex address 7168) → "

Note 4: " " (address specified by index register + hex 25) → "

ADC → Add with carry

$ → önündeki sayının hexsel formda olduğunu gösteri

(C) → eldenin içeriği

(A accumulator) → A akümülatörünün içeriği

hex 25 → 25 sayısı

(hex address 25) → 25 adresinin içeriği

Note 1

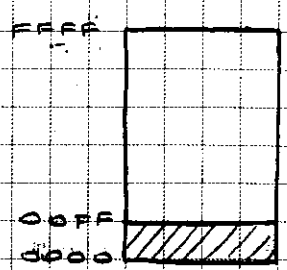
89	→ Emir
25	→ Sayı

Emir kodundan sonraki datalara immediate data denir. Program bellekte duran adreslere immediate adresleme denir.

Note 2

99	→ Emir kodu
25	→ Adres

Direkt adresleme



Yanım adreste bu sayfaya bakılır. Materialada bellek adresinin ilk sayfası register takımı için kullanılır. Genel amaçlı registerler için kullanılır, yani register olarak kullanılır.

Note 1 ve Note 2'deki adresler yanımdır. Note 3'te ise tam adrestir. (716F adresi)

Note 4 ADC A $25,X
↓
offset

25 adresi ile X'in (index) içeriği (belirttiği adres) toplanarak elde edilen adresteki değer A akümülatörü ile toplanıp elde değeri de eklenerek yanıp A akümülatörüne yazılır.

→ önündeki sayının immediate olduğunu gösterir.

JMP emri

JMP Address → Belirtilen adrese direkt atlanır.

Mnemonic	Explanation	Execution Time (cycles)	Introduction Code (Hex)	Condition for branch	Source Listing Example
BCC	Branch if carry clear	4	24	C=0	BCC $25
BEQ	Branch if equal to zero	4	27	Z=1	BEQ $25

100	
101	24
102	25

102'den sonra 25 adres ileriye atlar. yani 127'ye atlar.

ÖRN BCD ikili sayı dönüşümü yapılmak istenmiyor. BCD sayı iki basamaklı olup E000 adresinde tutulmaktadır. Dönüşüm kuralı aşağıdaki gibidir.

1- Tamsayı bölmesi yaparak BCD sayı ikiye bölünür.

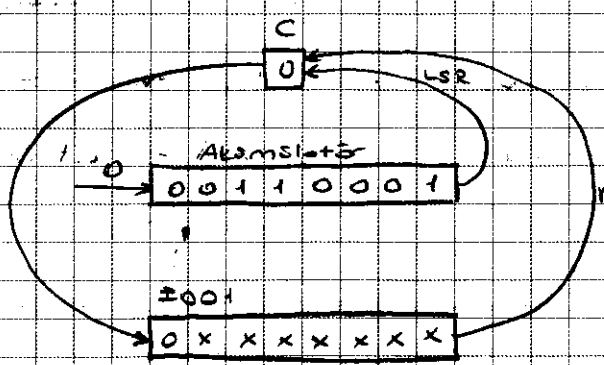
2- Kaydırmayla yapılan binner maddedeki bölme işleminde 1 olar basamağından 1'ler basamağına lojik-1 geçişi olmaz ise 1'ler basamağından 3 çıkararak düzeltme yapılır.

3- Kalan bir yerde saklanıp elde edilen bölüm tekrar tamsayı bölmesi yapılarak ikiye bölünür. Bu bölme işlemi bölüm basamağında tamsayı kalmayıncaya kadar tekrarlanır.

5- Kalanlar uygun sırada alınır.

Bu kuralı gerçekleyecek programı 6502 CPU emir takımını kullanarak yazınız. (Bu CPU'nun bölme emirleri yok)

```
    LDA  B#$08     BCD sayısındaki bit uzunluğunu belirle
    LDA  A$E000    BCD sayıyı A'ya taşı
(1) LSR  A         2'ye böl
    ROR  $E001     Kalanları sakla
    BIT  A#$08     } Biner basamağının MSB'si 1 mi?
    BEQ  (2)       } Yani bölme tamam mı?
    SUB  A#$03     Düzeltme yap
(2) DEC  B         Sayıcıyı 1 azalt.          } Bölme tamamlandı mı?
    BNE  (1)       Sonuç 0 değilse tekrar böl }
    WAI            Bekle
```



Bu örnekte B akümülatörünü sayıcı olarak kullanacağımızdan önce B değeri B akümülatörüne kaydırılır.

Onlar basamağı	Birler Basamağı	
0 1 1 0	0 0 1 0	= 62
0 0 1 1	0 0 0 1	= 31
0 0 0 1	1 0 0 0	= 18 (3 fazla çıkıyor)

8 bitlik sayı olduğu için 8 defa bölme yapılacak.

```
  x x x x   x x x x   = A
AND 0 0 0 0   1 0 0 0   = 08
  ---------------------
  0 0 0 0   1 0 0 0
```

Birler basamağından onlar basamağına 1 geçiş oldu mu?

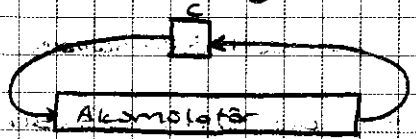
0 ise olmadı (*0 bayrağında tutulur)

1 ise oldu (sonuç 3 fazla, düzelt)

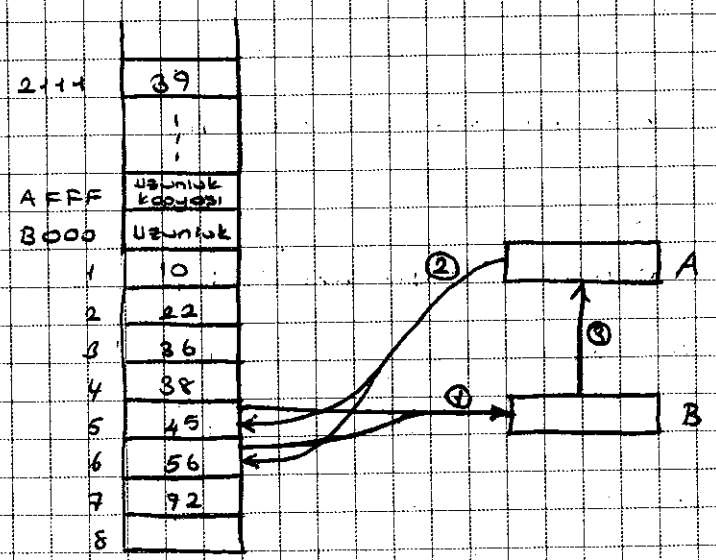
Eşitlik bayrağı olmayan bir CPU'da eşitlik testi yapılmak isteniyor. Eşitlik testi yapılacak veri akümülatöründedir. Bu veriden lojik-1 değerli bitlerin sayısının tek olması durumunda elde bayrağını setleyen bir programı 6502 CPU emir takımını kullanarak yazınız.

Yine B akümülatörünü sayıcı olarak kullanacağız.

```
LDA  B#$08      Bitlerin sayısını belirle.
CLR  Sayıcı     1 değerli bitleri sayan sayıcıyı(bel-
                lek bölgesi) temizle.
ROR  A          Sayıyı 1 bit sağa kaydırıp carry'e atar.
BCC  (1)        Elde 0 ise (1)'e git.
INC  Sayıcı     Elde 1 ise sayıcıyı 1 arttır.
(1) DEC  B      B akümülatörünün içeriğini 1 azalt. B
                sıfırsa sıfır bayrağı setlenir. Bu
                bayrağı kontrol et.
BNE  (2)        Sıfır bayrağı 1 değilse (2)'ye atla.
ROR  Sayıcı     Sayıcının en anlamsız bitine lojik-1
                değerli bitlerinin sayısının tek
                olduğunu göstereceğinden elde(c)
                gönderilir. Sonuca ulaşılır.
WAI             Bekle
```



Q2) B000 adresinden başlayan ve uzunluğu B000 adresinde tutulan kücükten büyüğe doğru sıralanmış ve elemanları 1 byte büyüklüğünde pozitif tamsayılar olan sıralı bir dizi veriliyor. 2111 adresindeki bir giriş cihazından gelen bilginin bu dizide bulunup bulunmadığını test ettikten sonra, eğer dizide yoksa dizinin sıralı olarak bozmadan uygun yere konulmasını sağlayacak ve dizi boyunu 1 arttıracak program 6802 CPU emir takımını kullanarak yazınız.



Uzunluk güncellenecegiinden eşitsel değeri kaybolmasın diye uzunluğun kopyasını alıp AFFF adresine kaydık.

```
        LDX  #$B001     Dizinin başını belirle.
        LDA  B$B000     } B000 adresindeki dizi uzunluğunu
        STA  B$AFFF     } AFFF ye yaz. (kopyasını al)
        LDA  A$2111     Dışarıdan bilgiyi gir.
①       CMP  A$00,X     Akümalatör - Bellek (sonucu Akü'ye kaymaz)
        BEQ  SON        Akü - Bellek 0 ise bu eleman dizide var-
                        dır. Hiçbirşey yapmadan sona git.
        BMI  YERLES     Akü<Bellek ise yerleştir
        INX             Akü<Bellek değilse indexi arttır.
        DEC  $AFFF
        BNE  ①          1 arttırıp karşılaştırmaya devam et.
YERLES  LDA  B$00,X ①   } Sayının konula-
        STA  A$00,X ②   } cağı yer belir-
        TBA         ③   } lenmiştir. Kay-
        INC             } dırma yap ve
        DEC  $AFFF      } sayıyı belleğe
        BGE  YERLES     } yerleştir.
        INC  $B000      Dizi boyunu 1 arttır.
SON     WAI             Bekle
```

Örn: İşaretsiz ikili sayı dizisi azalan sırada tasnif edilmek isteniyor. Dizi uzunluğu 0040 Hex. adresinde tutulmaktadır. Dizinin kendisiyse 0041 Hexi adresinden başlamaktadır. Bu sayı dizisi tasnif programını 6802 CPU emir takımını yazınız.

Örnek dizi: 25 12 86 69 10 70

```
    LDA   A#$00      } Takas bayrağını sıfırla.
    STA   A$0038     }
    LDA   A$0040     } Dizinin uzunluğunun kopyasını
    STA   A$0039     } çıkar.
    DEC   $0039
    BEQ   (4)        Dizi uzunluğunu azaltınca 0 ise (4)'e git.
    LDX   #$0041     Dizinin başlangıcını belirle
(1) LDA   A$0,x      İlk değeri A'ya kay
    CMP   A$1,x      Komşu iki sayıyı karşılaştır (A<=B
                  (2)
    LDA   B$1,x      } Küçük sayı önde ise takas yap.
    STA   A$1,x      }
    STA   B$0,x      }
    LDA   A#$01      } Takas yapıldığını hatırla. (while)
    STA   A$0038     }
```

② INX
DEC $0039
BNE ①
DEC $0038
BEQ ③
④ WAI

} Dizinin sonuna gelindi mi?

} Yeni tarama gerekiyor mu?

0039	Bizi uzunluğu için kopyası	
0040	Bizi uzunluğu	
1	25	
2	12	
3	56	
4	69	
5	10	
6	70	

0041 IR

*Sakın akümülatöre 16 bitlik değer kayma. Çünkü CPU'muz 8 bitlik.

Adres	Makina kodu		
2000	86	00	...
2002	B7	00	38
2005	B6	00	40

```
do
{
    for(_____)
    {

    }
}
while(___)
```

ÖRN: Bir mikrobilgisayarda pozitif sayılar işaretli genlik, negatif sayılar ise ikili tümlenmiş biçimde tutulmaktadır. A000 adresinden başlayan, her biri 1 byte uzunluğundaki sayılardan oluşmuş 256 bytelik bir dizi veriliyor. Bu dizideki sayılardan genlik değeri en büyük olanı bulup E000 adresine koyan bir programı 6802 CPU emir takımını kullanarak yazınız.

```
    LDA   A#$FF      } Dizinin boyunu belirle.
    STA   A$1000     }
    LDA   B#$00      En büyük sayı 00 olsun.
    LDX   #$A000     Dizinin başlangıcı
(3) LDA   A$00,X     Dizinin ilk elemanını çağır.
    BIT   A#$80      Sayı negatif mi? (İşaret bayrağına bakılır)
    BPL   (1)        İşaret bayrağına bakılır. D- (1)
    NEG   A
(1) CBA              En büyük sayı ile ilk elemanı karşılaştır
    BMI , (2)        [A]-[B] = (-) ise (2)'ye git.
    TAB              En büyük sayıyı B akümülatöründe tut
(2) INX              Indexi 1 arttır.  } Dizinin sonuna
    DEC   $1000      Boyu 1 azalt.     } gelindi mi?
```

BGT ③

STA B $E000

WAI

} en büyük sayıyı sakla.

1	0	1	1	0	1	1	0

↑ işaret biti

B akümülatöründe en büyük sayıyı tutuyoruz.

0 1 1 0 1 = +13

↕

1 0 0 1 1 = -13

↓

İşaret biti 0 ise pozitif,

İşaret biti 1 ise negatif. Sayının ilk biti aynı kalır, diğer bitlerin tümleyeni alınır.

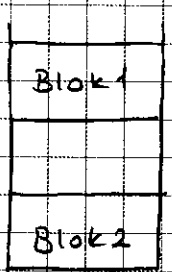
I.VİZE SORULARI

① b

A

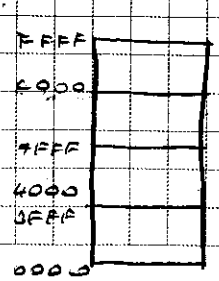
I

I

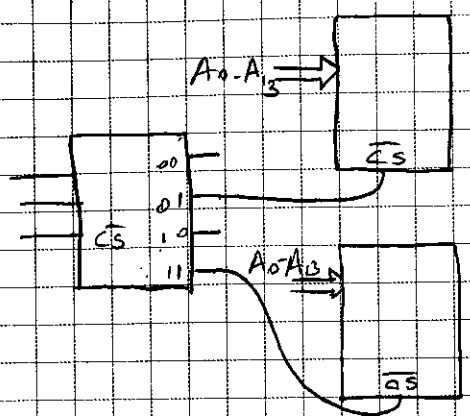


Index fiziksel bize programımızı blok olarak taşımamızda yarar sağlar.

②

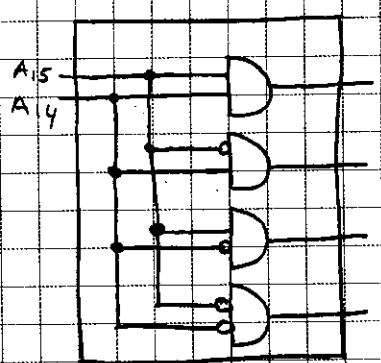


16K * 8 bit



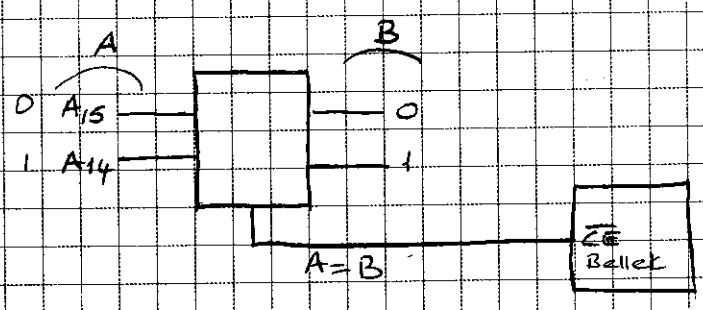
A15	A14
0	0
0	1
1	0
1	1

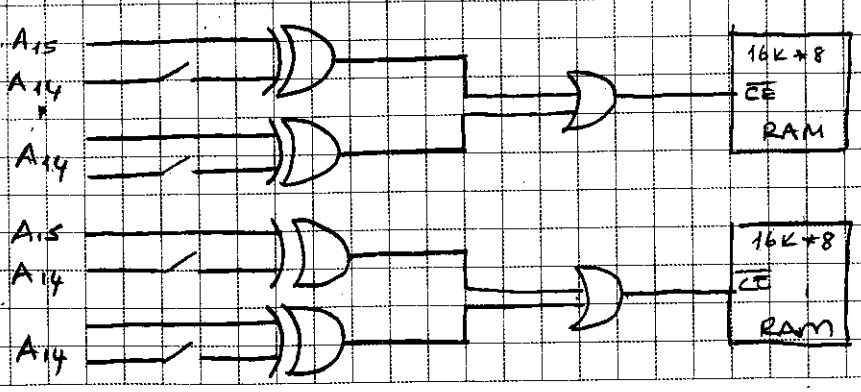
a



Tek katlı AND örgüsü

b



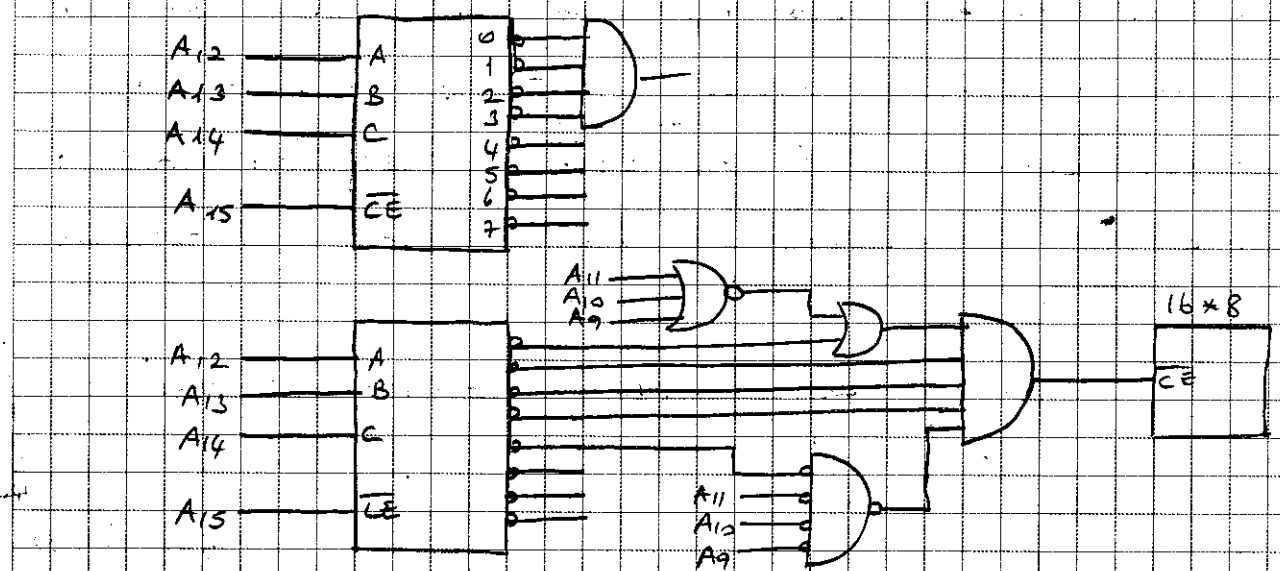


$A_{15}\ A_{14}\ A_{13}\ A_{12}\ A_{11}\ A_{10}\ A_9\ A_8$

C $0200 = 0000\ 001x\ xxxx\ xxxx$

$= 0011\ 11xx\ xxxx\ xxxx$

$4200 = 0100\ 0010\ 0000\ 0000$



③ LDA A #$ 2E

LDA #$ 0041

② CMP A$00,x

BEQ ①

INX

DEC $40

BNE ②

① LDA A#$20

③ STA A$00,x

INX

DEC $40

BNE ③

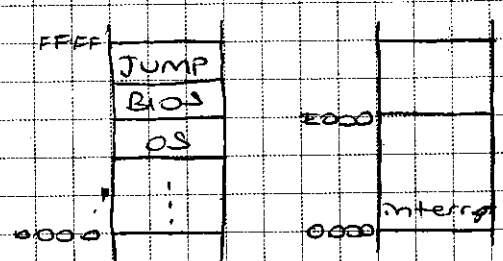
WAI

Z-80 için:

Bir bilgisayar sistemi kurar. OS'sinin de nasıl aktif yapılacağını düşünecegiz. 8 bitlik CPU larda OS denilen kısım monitor programıdır ve gerilim verildiğinde bu program girilmelidir. Bunun için bellekte 0000 alanına oturmalı ve bilgisayarda bu adresi üretecek düzenek olmalı.

Z-80'de OS 0000 adresinden başlar çünkü 0000 da BIOS oturur. Onun için bu adresi otomatikman üreten bir programımız veya lojik yapımız olmalı.

Pclerde resette FFFF üretilir ve BIOS FFF0'a oturur. Jump emri ile BIOS'a atlarız sonra OS'ye geçer.



⇒ Interruptların bulunduğu adres CPU'nun yapısından gelir.

→ Devredeki etiketler

① Yani reset verildiğinde adres busunda F000 adresini üretecegiz önce.
Tuşa basıldı akım aktı, kondansatör boşaldı monostable multivibratörü tetiklendi Q çıkışından 1 üretti, Q ise 0 oldu ve reset üretti. Yeniden tetiklenebilir türü değildir, * ② Flip-flioplarda asenkrondur clocktan bağımsızdır, reset ve 'clear.

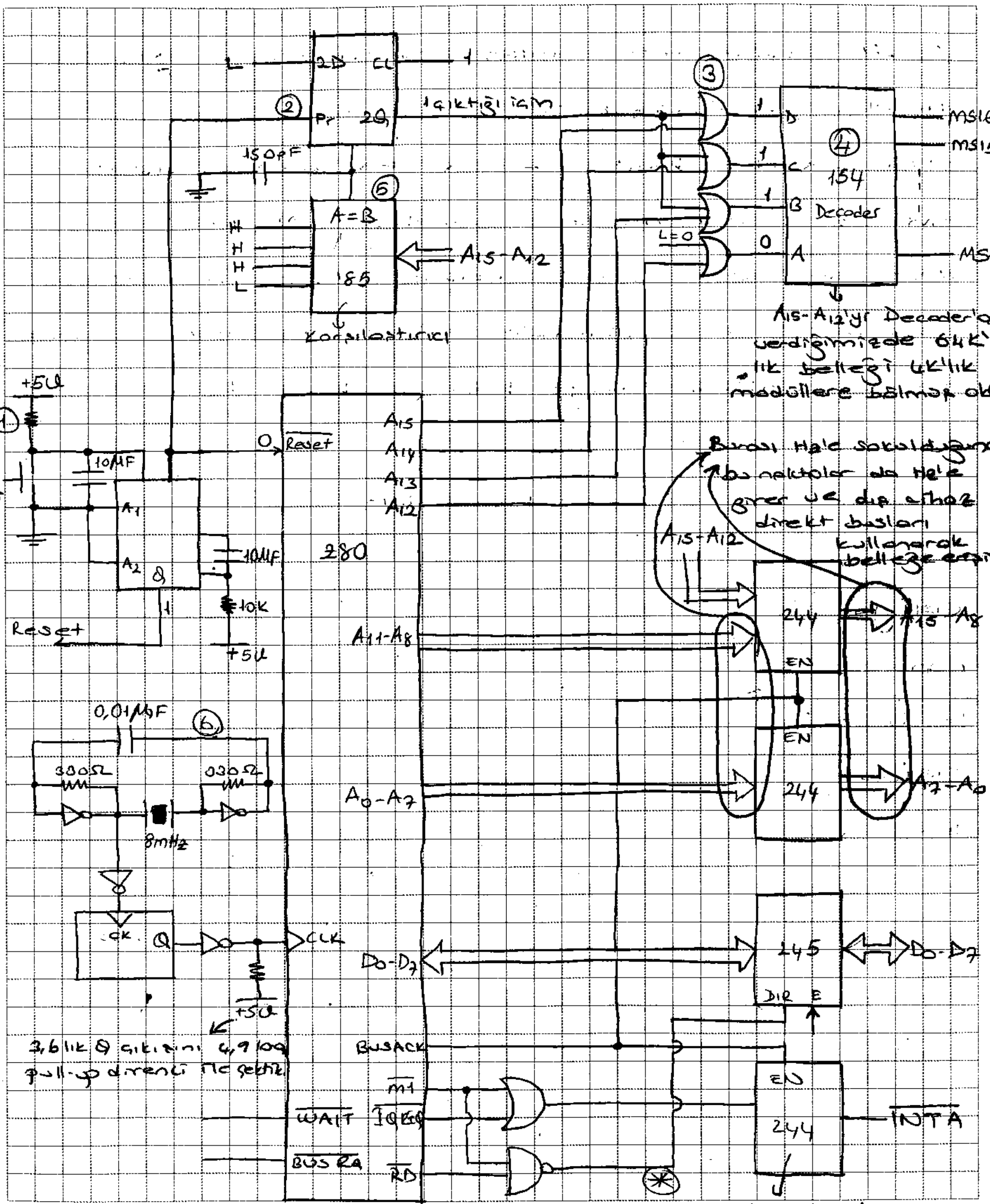
③ Eşit sistemi devresi

④ Decoder 2=0 olduğu için D=0

DM data transferinde dış cihaz kendi datasını üretir, CPU bu işte görevli olmaz.

⑤ Eşitlik karşılaştırıcı görevinden yararlanacağız. '85' in girişi 1110 A_{15}-A_{12} 1110'sa A=B uçundan 1 çıkıyor, HHHL ile A_{15}-A_{12} yi karşılaştırıyor, Eşitse CK=1 gelir ve D'deki giriş Q'ya aktarılır yani 1 Q'ya aktarılır, çıkış 0 olur.

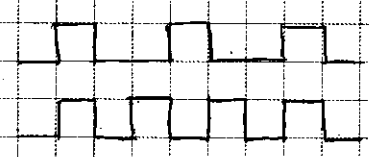
* Bizim FFlo verdiğimiz F000 adresi geldiğinde CPU orayo yazmış olduğumuz JUMP emriyle F003 uçuruz ve PC'nin yani CPU'nun F adresi üretir hale gelmesini sağlarız ve biz F000 adresi vermeyi durdururuz.



Bufferlamak için koyduk.

150 µF kondansatör gelen işaretteki gürültüyü, dalgalanmayı azaltır.

(6) CPU saatle kontrol edilen bir devre olduğu için bize saat devresi gerekir. 8 MHz'lik bir saat devremiz var. Yüksek seviye süresi ile alçak seviye süresi birbirinden farklı ve biz bu frekansı 8 MHz'lik ikiye bölersek kare dalga elde ederiz.



CPU'ya yeni bellek kartı eklemek gerektiğinde bunun ne olduğunu bilmediğimiz için CPU yanabilir. Onun için buslara buffer koyarız.

CPU'dan hızlı cihazlar olduğunda direkt buslara istememiz gerekecek bunun için $\overline{BUSRQ}$ işareti kullanılır. Bu aktif olduğunda bufferları devre dışı bırakmak için bir enable işaretine ihtiyacımız vardır. Bazı cihazlar yavaş çalışmalarına rağmen performansı arttırmak için bu buslara isteyebilir. BUSRQ işareti geldi. BUSACK kullanılarak bufferlar disable yapıldı. CPU da bacaklar yetmediğinden Acknowledge

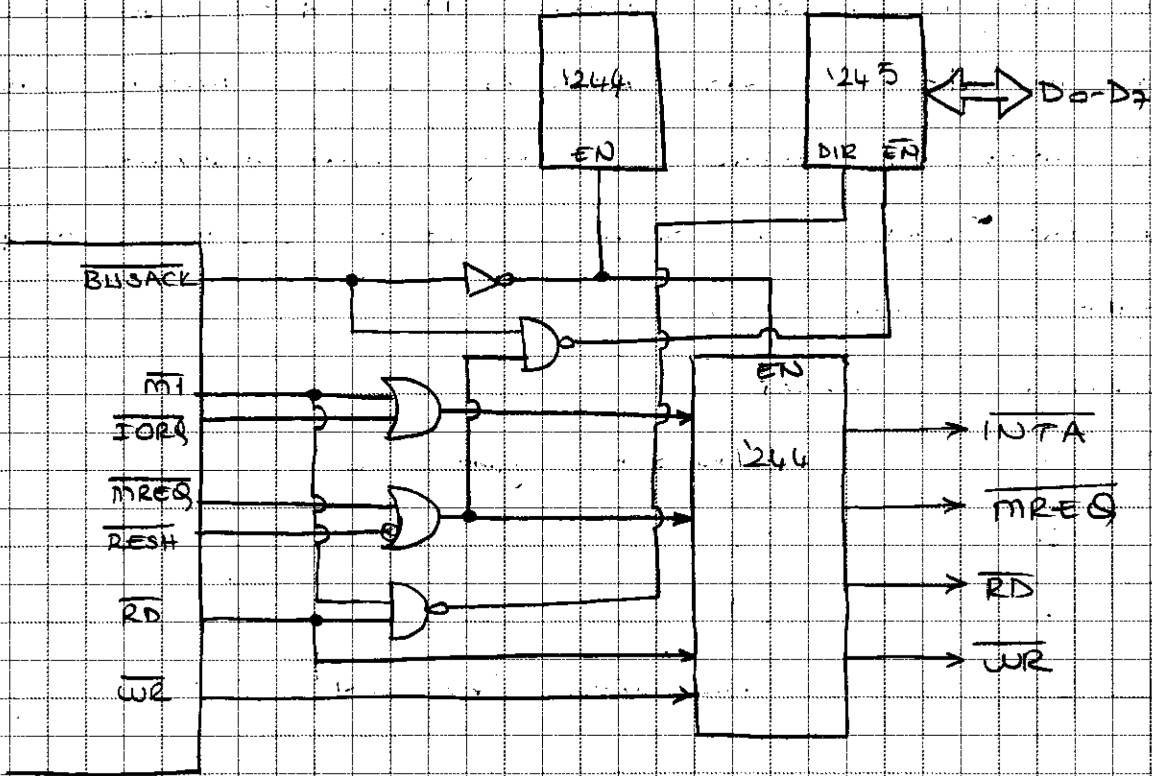
İşaretler interrupt işaretini kendi üretir.

Z-80 Emir kodu alımından sonra dinamik bellekleri refreshlenmesini sağlayan CPU'dur. Emir kodu alımı sırasında refreshleme yapıldığı varsayılır. Emir kodu alımı kısa sürdüğü için bellek devresi yavaş olduğundan M1 ile emir kodu alımını gösterir ve emir kodu alımında belleğe erişmek için ayrılan o süreyi uzatmak için kullanılır. IORQ giriş çıkışa erişimi gösterir. Bu iki aynı yerden bir anda bir tanesi aktif olduğu için (yani ikisi aynı anda aktif olamaz) iki işaretin kombinasyonundan Acknowledge işareti üretilir. (*)(**)RD neden M1 ile yönlendirilip bir oldu? Çünkü interruptlar onaylandığı zaman interfelde CPU bir identification datası bekler, Amaç bu datayı CPU'ya göndermek. Interrupt Acknowledge bellekten okuma yapmadan CPU'nun dışarıdan data alması gerektirir. Bunu RD ile M1 birleştirerek sağlarız, yani bellek kullanılmadan identification data alımı, interrupt

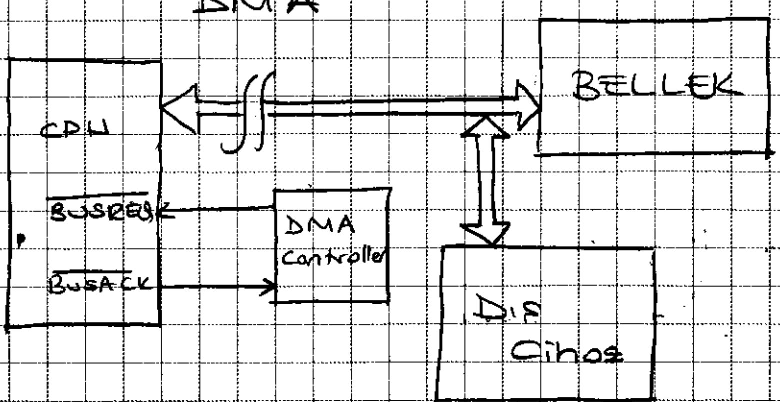
anayloma anında emir kodu alınmayacak da identification data alınacak. M1 emir kodu alınaçağı zaman aktif olur. Bir wait komutu kullansaydık bunun için zaten daha alımı uzun sürer, bir de diğer bellek okuma ve yazma için uzatma yapmamak için M1 komutu kullanıyoruz (beklememek için). Bir de Interrupt Ack anında aktiftir. Zaten 2-80 'in böyle int Ack için özel bir uçu yoktur. Sadece emir kodu ve daha alımı anında değil de interruptta da CPU içeriye yönlensin diye kullanılır.

Dinamik belleklerin refreshlenmesi için emir kodu alımı sırasında bekletme yapılıyor ve bu sırada buslar boş kaldığı için bellek refreshlenir. Bu yapıda iki görev bir makine periyoduna sıkıştırılmış olur. Emir kodu alımına ayrılan süre daha okuma süresinden kısadır. İkisi de daha okuma olduğu halde bu yüzden emir kodu alımında wait kullanırsak busdaki işaretlerin tamamı için uzatma yapılır ama biz sadece emir kodu alımındaki süreyi uzatmalıyız diğer busları boş kalsın ve okumada refreshleme yapılsın

diye. Normal okuma işlemi uzundur CPU yeteri kadar zaman ayırır zaten, M1 ile emir kodu alımı süresini uzatırız.



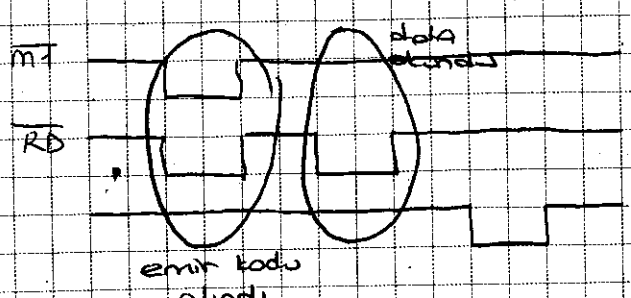
DMA



Data controller BUSREQ ile CPU'ya giden adres ve data bağları kopadı. Böylece bellek ve dış cihaz arasında data transferi olacak bu da data controllerin bu iki cihazı kontrol etmesiyle sağlanacak, CPU hiçbir işlem yapmayacak. Bu duruma direct memory access (doğrudan bellek erişimi) (DMA) diyoruz.
Refreshleme anında memory request olacak olmayacak (belleğe erişilmez)

$\overline{IOREQ}$	$\overline{MREQ}$	
0	0	Data bus yüksek-z
0	1	} Data bus enable yapılmış
1	0	
1	1	Data bus yüksek-z

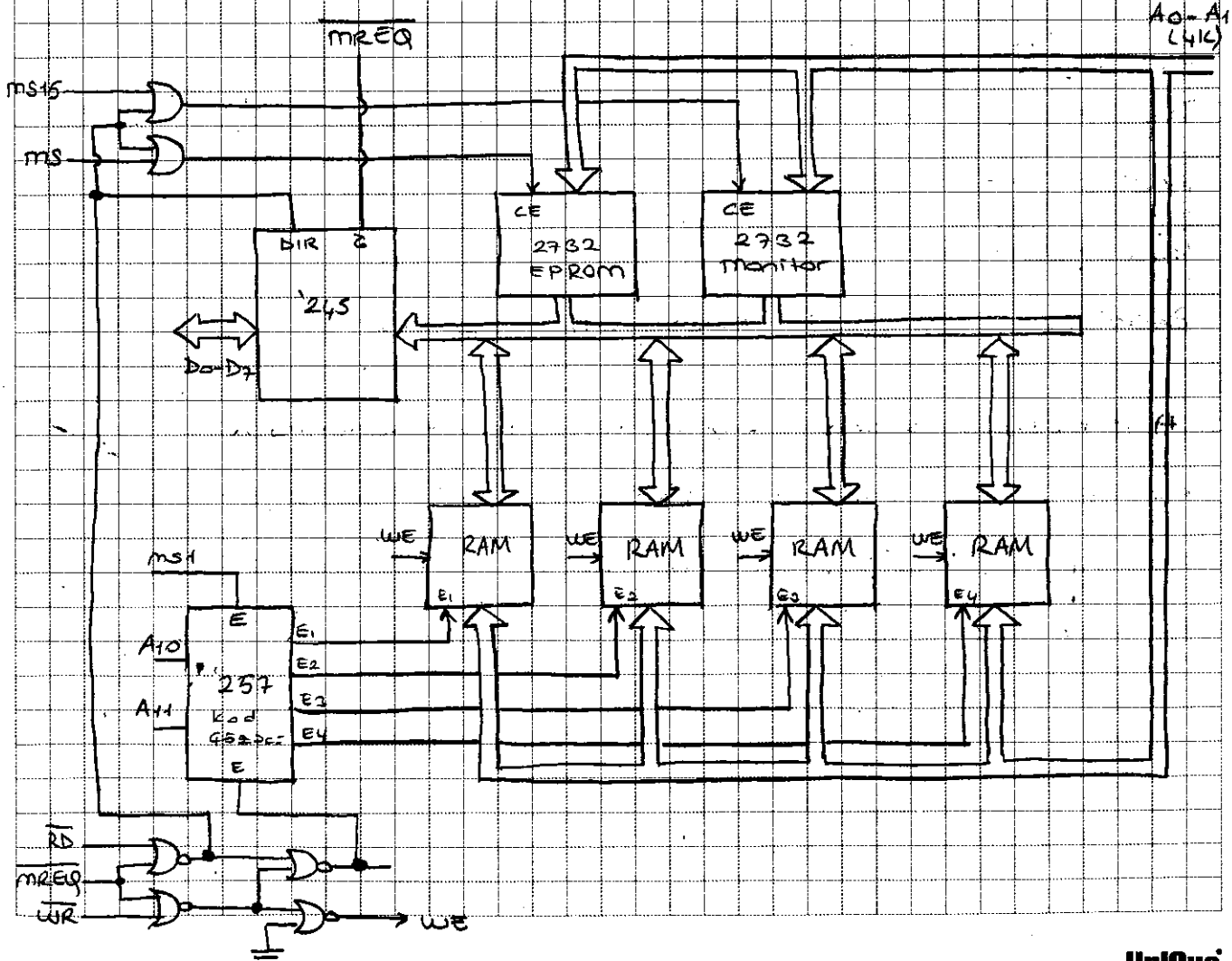
$\overline{M1}$ → Emir kodu alındığını gösterdiğinden her emir kodu alınırken aktif.



M1'i $\overline{RD}$ ile birlikte kullanıyorsa çünkü emir kodu mu, data mı alınacağını belirtmemiz gerekiyor. M1 emir kodu aldığımızı gösteriyor.

M1 aynı zamanda refreshlemede kullanılır. Emir kodu alımıyla refreshleme aynı makine periyodunda yapılır.

Bellek ve Giriş-Çıkış Kapılarının Tasarlanması

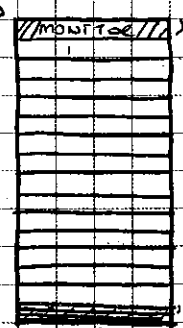


EPROMlar 2708den başlarlar (1K) ..... (10bit = 1K)

2716 (2K) 000 / 400 } 1K

2732 (4K)

E000



16
x 4
64K

M16 ile M8'yi kaplayacak yerleştiriz.

A_{12} A_{13} A_{14} A_{15}

Diyelim ki 1K*8bitlik RAMleri buraya koyacağız. Adres uçlarından 2 tanesi açıkta kalmıştı. Bu uçlarla kod çözeriz.

Bellek okumada gereken DIR ile önceki farklıdır.

Aritmetik İşlem kartının Tasarlanması

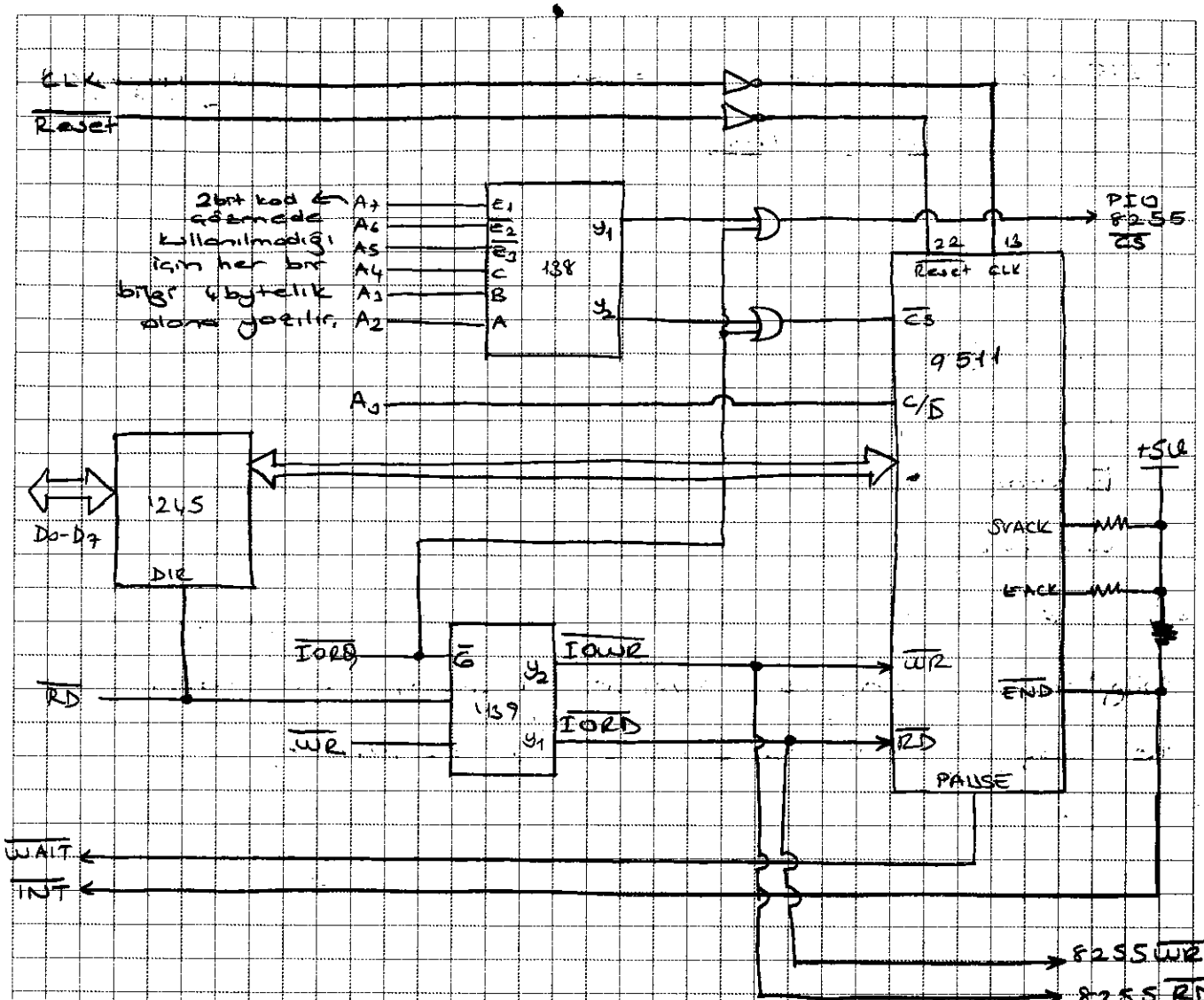
* C mi daha çok yer kaplar yoksa nesne kullanarak yaptığımız program mı? - C

* '139 ve '138 nedir?

↓ multiplexer gibi

↓ 3-to-8 kod çözücü

IN A, port } 8bit
OUT port, A } Emirler



8255 → Bunumla da kullanılmayan paralel giriş-çıkış yapmaya yarayan eleman

1000 10xx

+ 8

88 ile 8B arasındaki bölgede durur

- $\overline{WAIT}$ → CPU'nun verdiği zaman I/O cihazına yetmez ise I/O cihazı CPU'ya $\overline{WAIT}$ gönderir, CPU 1 saat peryodu bekler.
- $\overline{INT}$ → İşlemler tamamlandığında bu uçtan CPU'ya interrupt gönderilerek bildirilir. CPU da bu datayı I/O biriminden alır.
- C/D (komut/data) → Veri gönderilirken A_0 1 sıfır, komut gönderilirken 1 yapılır.
- 9511 de registerlar stack türü registerlardır, yani bilgiler hep en üste yazılır, okudukça da veriler üstteki datayı çıkar. Adresleme gerektirmez.

GİRİŞ/ÇIKIŞ (INPUT/OUTPUT)

A. Paralel Giriş-Çıkış: 1. Programlı G/Ç
2. Kesmeli G/Ç
3. DMA'lı G/Ç

B. Seri Giriş-Çıkış: 1. Asenkron G/Ç
2. Senkron G/Ç

Paralel giriş-çıkış kullanacaksak I/O cihazımız bilgisayara yakın olmalıdır. Çünkü paralel giriş-çıkışta datamızın her biti için ayrı hat kullanırız. Bu da 8 bit için 8 hat demektir. Bu 8 bitlik dataya ilaveten 1 bit Data Ready, 1 bit Data Request ve 1 bit Ground olmak üzere en az 11 bitlik yol kurmalıyız. Uzak cihazlarda seri giriş-çıkış kullanmak daha uygundur. Çünkü seride paraleldeki 8 iletken yerine 2 iletken yeterlidir. (Biri giriş diğeri çıkış için). Eğer cihaz çok fazla uzak değilse hızı arttırmak için Data Request, Data Ready, Ground, vs. uçlarını paralel bağlayabiliriz. Telefon hattı gibi uzun hatlarda bu bilgiler de seri olan 2 hattan

gönderilir.

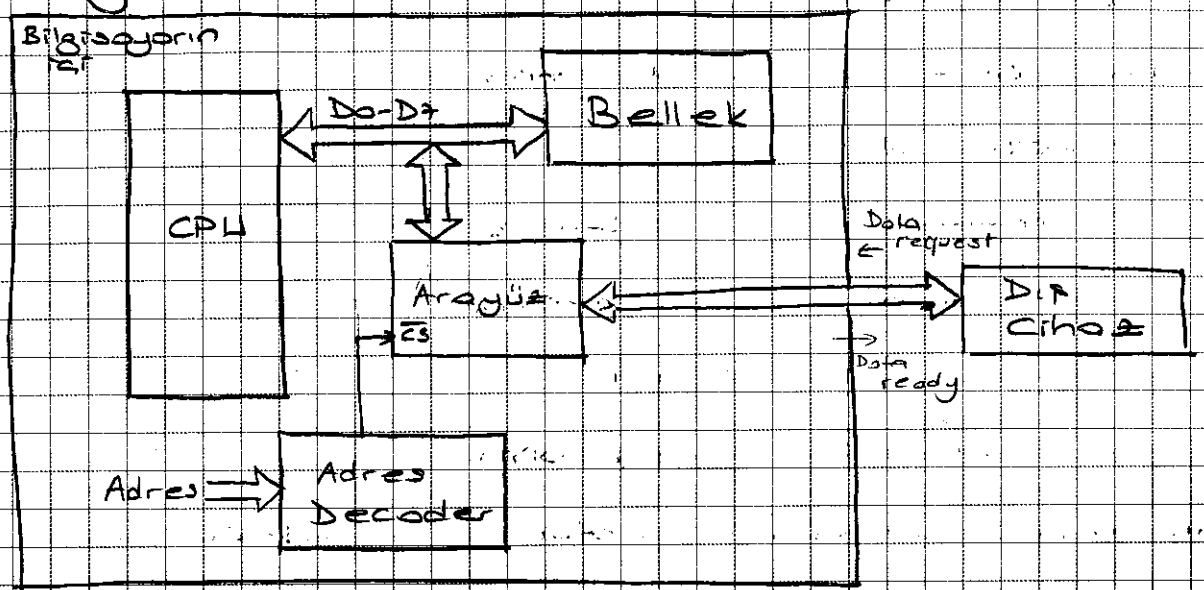
1- Programlı Giriş-Çıkış: Data transfer isteğinin algılanması ve data transferi programla gerçekleştirilir. En ucuz ama en yavaş yöntemdir.
CPU sürekli olarak data request'i kontrol eder. İstek gelmediği sürece CPU bu işi sürekli kontrol ettiği için CPU'yu gereksiz meşgul etmiş olunur. Bu da hızı düşürür.

2- Kesmeler (INT) Giriş-Çıkış: Data transfer isteğinin algılanması donanımla, dataların transferiyse programla gerçekleştirilir.
Gerekli donanım hem CPU'da hem de giriş-çıkış cihazının bağlandığı portta bulunur. İstek gelmediği zaman CPU başka işlerle uğraşabilir. Donanım kullanıldığından maliyet artar.

3- Doğrudan Bellek Erişimli (DMA) Giriş-Çıkış: Bilgisayarımız yavaş çalışan CPU'lu olarak algılandığından dış cihaz hızı beğenmiyor ve dış cihaz belleğe CPU'dan bağımsız DMA cihazı yardımıyla doğrudan erişiyor. Data transferi de tamamen donanımla yapılıyor. En hızlı ve en pahalı yöntemdir.

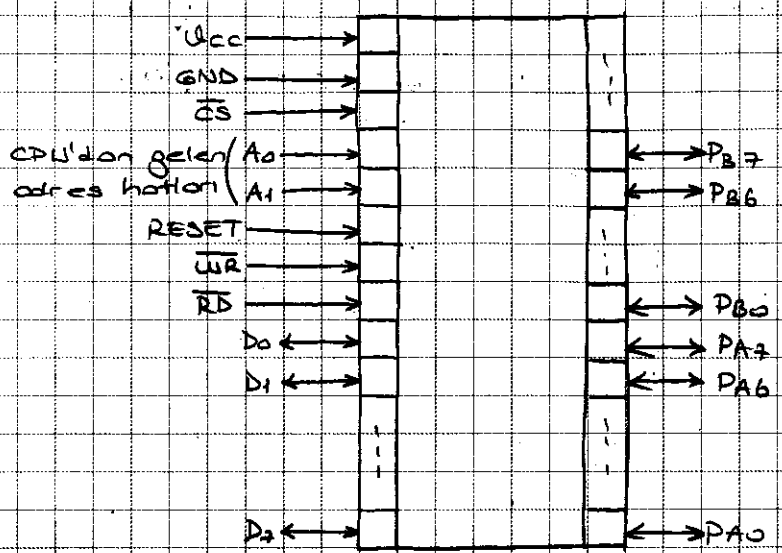
*Bu üç yöntem tüm bilgisayarlarda bulunur.

① Programlı Paralel Giriş-Çıkış



Her dış cihaz elektriksel karakteristikler (DC karakteristik) bakımından bizim CPU muza uygun olmayabilir. Bu sebeple arayüz kullanılır. Dış cihaz seriyse arayüz seri, paralelse arayüz paraleldir. (378, 379, 37A printer arayüzleridir.)

Arayüzün yapısı:

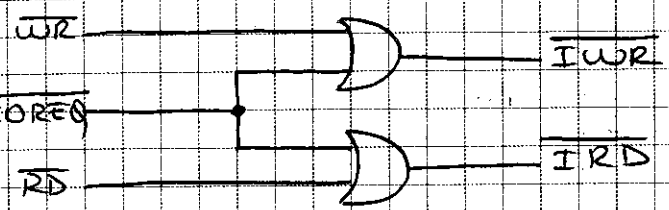


CPU arayüz cihazını seçmek için adres üretir. Bu adresin bir adres kod çözücüde kodu çözülerek arayüz cihazının $\overline{CS}$ ucuna yapılmış olur ve arayüz cihazı seçilir.

Dış cihaz ve arayüz $\overline{WAIT}$ üretmez. Sistem tasarımcısı ikisinin hızını da bildiğinde ekstra bir lojik devre tasarlayıp sisteme ekler bu lojik $\overline{WAIT}$ üreterek CPU'yu gerekli saat periyodu kadar bekletir.

Arayüz cihazının içindeki registerleri seçmek için ise A0 ile A1 uçları kullanılır.

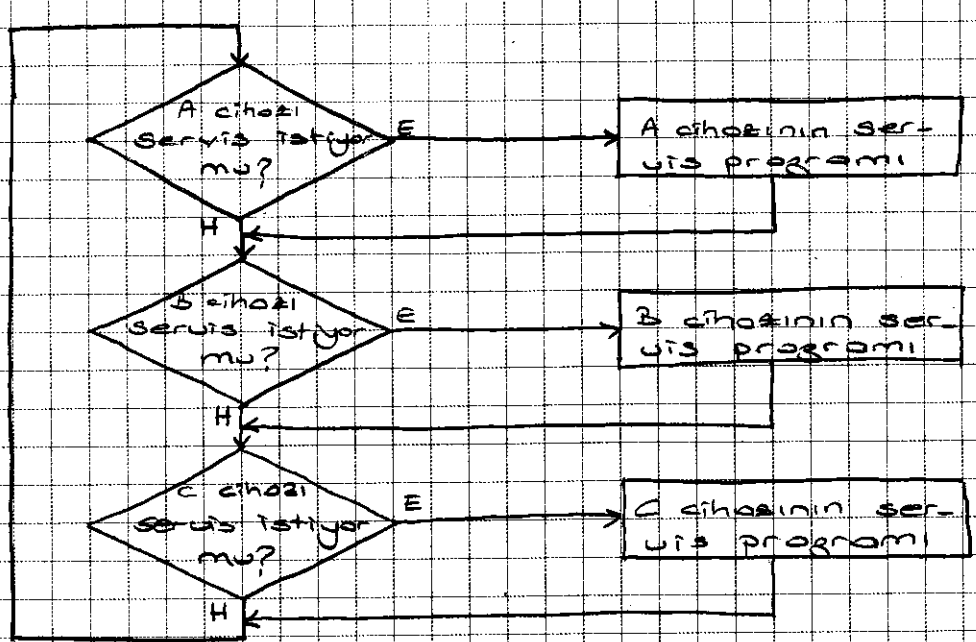
CPU bir arayüz cihazın seçtiğinde ya okuma ya da yazma yapar. Bunun için de $\overline{WR}$ ve $\overline{RD}$ yolları interrupt veri ($\overline{IOREQ}$) ile orlanır.

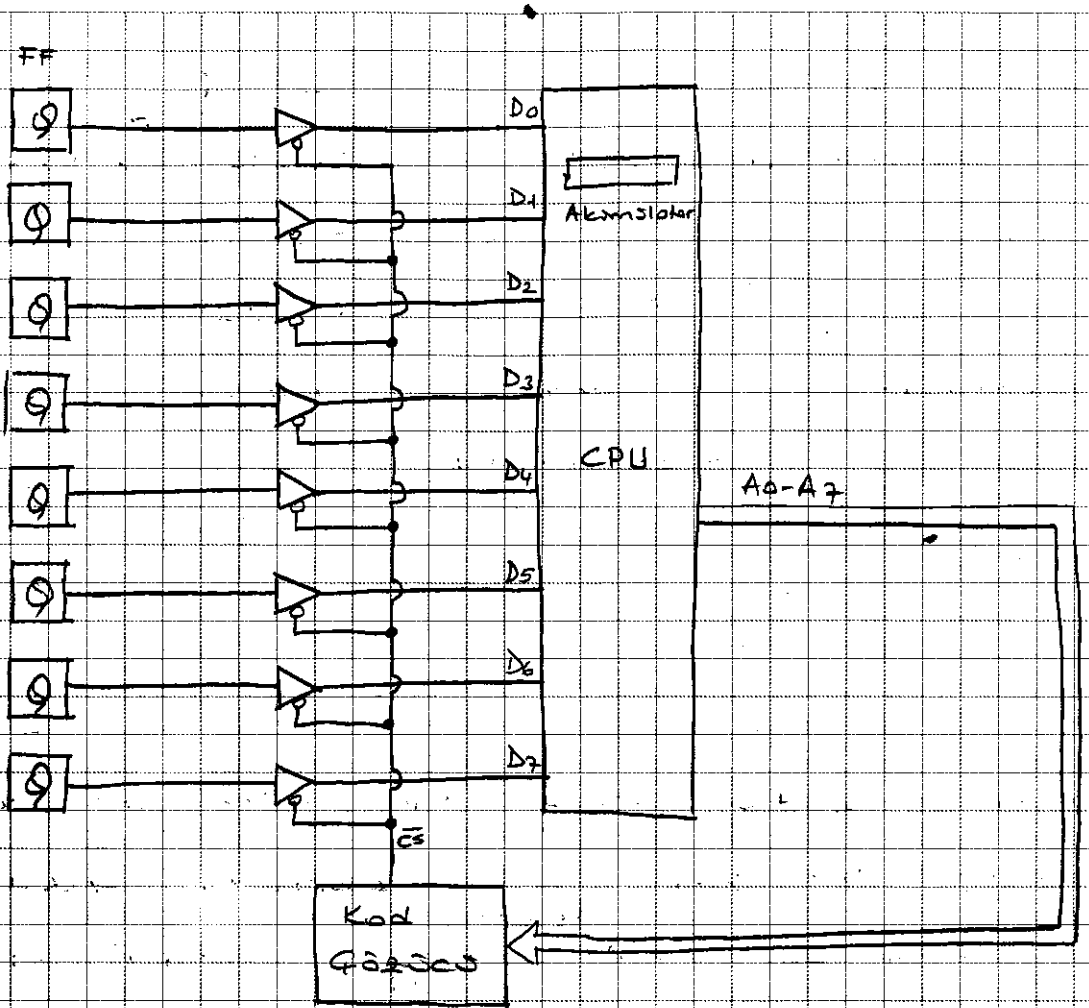


Bellek dışında tüm cihazların reset veri yordur. CPU bu reset yolunu seçerek cihazları resetleyebilir.

Çok cihazlı Program giriş-çıkış

Polling (yoklama) döngüsünün akış diyagramı

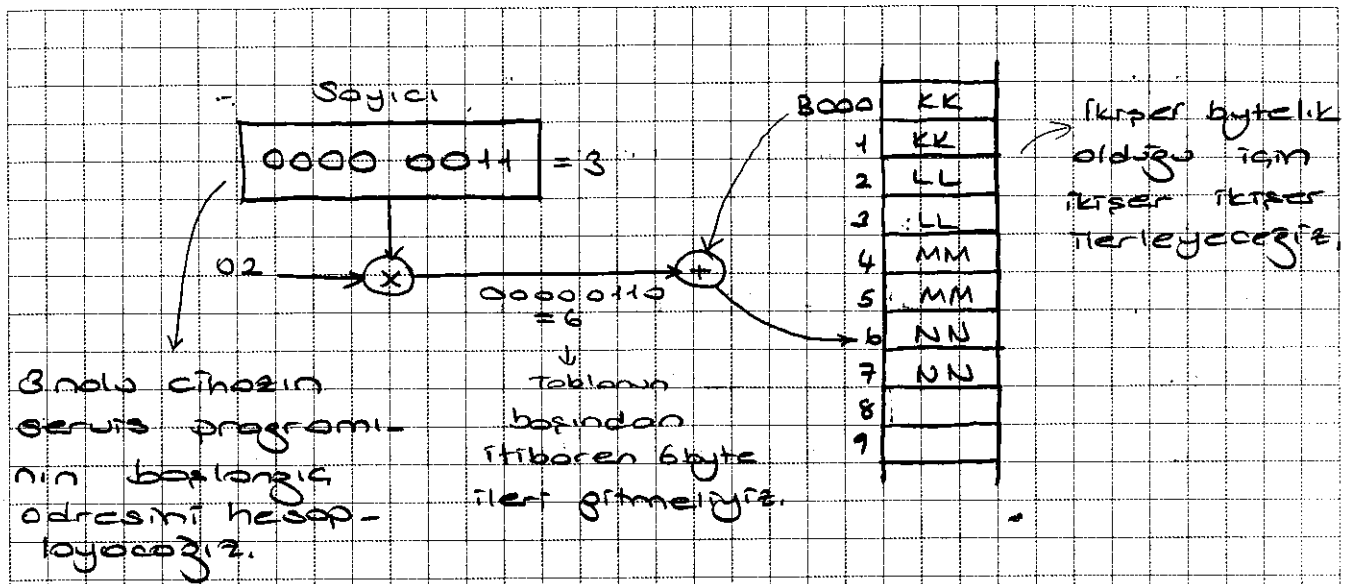




04 Sayıcı

İstek yoksa 1 arttırarak gidiyoruz. Diyelim ki: 4 oldu- ğunda istege rastladık. Bu bize 4 nolu cihazın istek- te bulunduğunu gösterir.

Çok cihazlı giriş-çıkışta sürüş programlarının boş- lanğıç adresleri bellekte bir tablo halinde tutu- lur.



NNNN adresini program counter'a yüklemek için bu adres önce SWI (Software Interrupt) kalıbının baktığı FFFA-FFFB adresleri arasına konulur. Daha sonra SWI komutu uygulanarak FFFA adresinin içeriği program counter'a yüklenir.

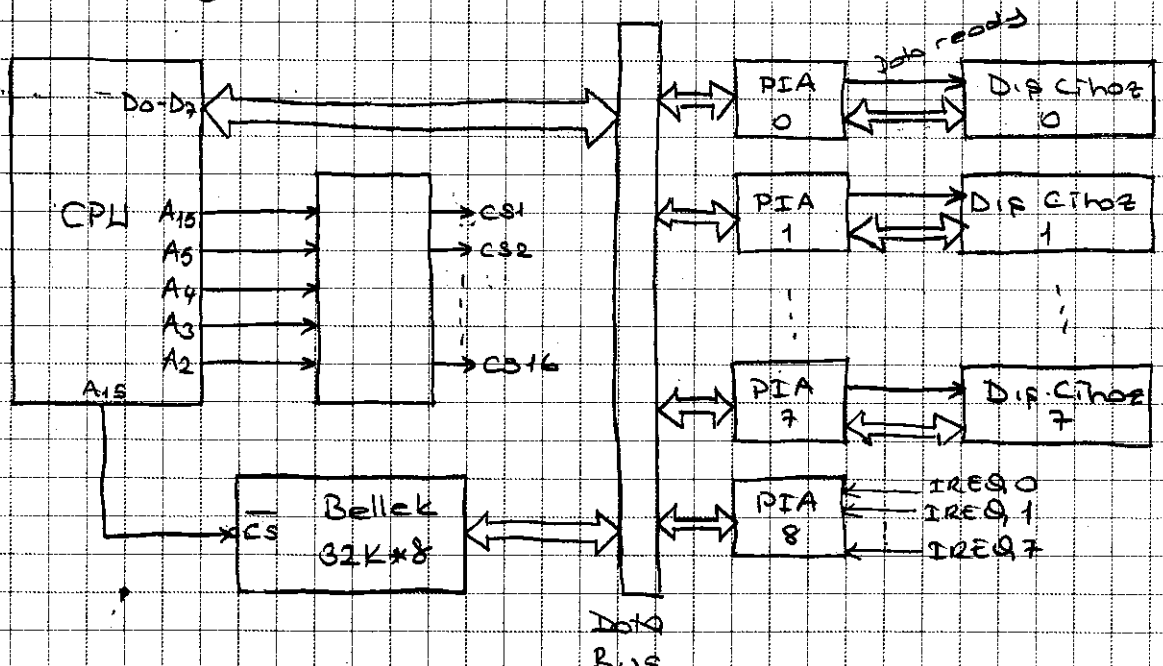
ÖR: Programlı giriş-çıkış yöntemi kullanılarak 8 farklı dış cihaza veri transferi yapılmak isteniyor. Her dış cihaz ayrı bir paralel giriş-çıkış elemanına (PIA) bağlanmıştır. Veri transfer isteği de ayrı bir PIA üzerinden CPU'ya iletilmektedir.

a) Dış cihazların veri transfer isteklerini yok-

lama (polling) yöntemiyle en kısa sürede test edecek bir donanım içeren CPU, bellek ve PIA lerden oluşan bir mikro bilgisayar tasarlayacak çalışmasını anlatınız. Bellek 30KB büyüklüğünde olup bellek haritasının 2. yarısını işgal etmektedir.

b. Veri transfer isteklerini test etmekten sonra en yüksek öncelikli dış cihazın servis programının başlangıç adresini belirleyip program counter'a koyan bir program 6802 CPU emir takımını kullanarak yazınız.

a)



2. yarıya ilişkin bitler için Adres kod çözme işlemi yapılmaz.

PC 256 tane cihaz kabul edebilir. 16'sı hard-ware, diğerleri software.

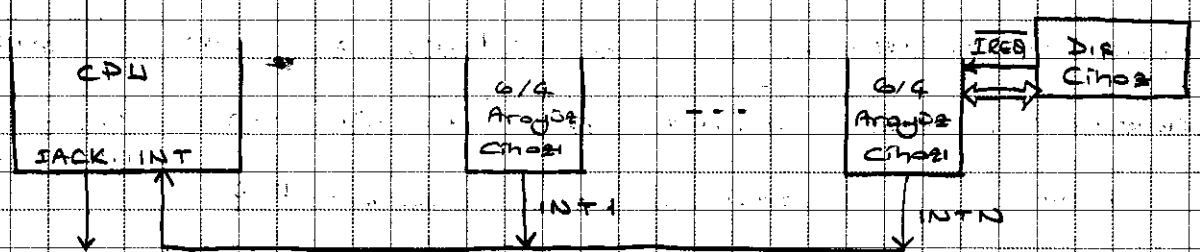
000
400) Aralığındadırlar.

b. LDX #$A000 } Servis programlarının başlangıç adres tablosu.
STX $FFFA }

④ CLR B Sayıcıyı sıfırla.
LDA A$PIA8 Tüm istekleri okunmalılara al. (oku)
② ROR A A'yı kendi üzerinden döndür. } İstekleri test et.
BCS ① İstek varsa 1'e git.
INC B Sayıcıyı 1 arttır. } İstekleri test etmeye devam.
CMP B#08 8'e eşit mi bak.
BNE ② Eşitse ②'ye git (başa dön)
JMP ④ Değilse ④'e git.
① ASL B Sayıcıyı 2 ile çarp.
STA A$FF00
LDA A$FFFB
ABA
BCC ③
INC $FFFA
③ STA A$FFFB

```
LSR B
LDX $FFFA  ⟶ A008 → IR
LDX $00,X  ⟶ XXYY → IR
STX $FFFA  ⟶ XXYY → FFFA,B
SWI        ⟶ XXYY → PC
LDX #$A000
STX $FFFA
LDA A $FF00
JMP (5)
```

② Kesmeli (Interrupt) Giriş-Çıkış

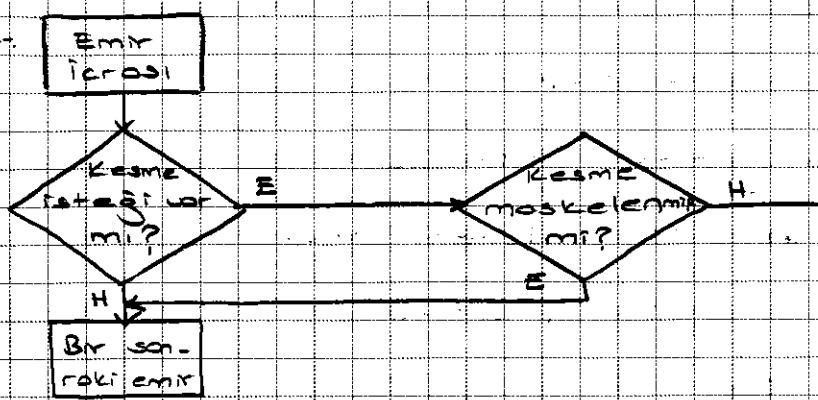


Birçok cihaz aynı anda kesme isteyebilir.

KESME Listesi

- Programlı giriş-çıkış senkron bir çalışmadır.
  Kesmeli giriş-çıkış asenkron bir çalışmadır.
- Çıkışlar open collektörlü olduğundan yonmaz. Bir molde totempole de yonacağından çıkışlar aynı yere bağlanmaz.

UniQue design

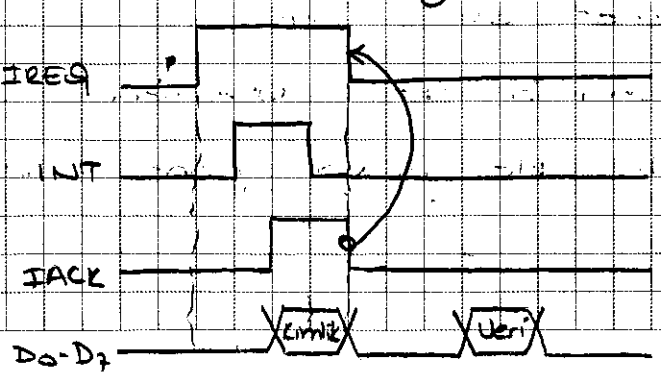


KESME LOJİĞİ

Her emrin son saat periyotlarında kesme işareti alıp almadığı kontrol edilir. Böylece emrin icrası yarıda kalmaz.

Maske kesme işaretinin değerlendirilip değerlendirilmeyeceğini belirten bir bitlik bir işarettir. Maske varsa bu tür emirler CPU tarafından karşılanmamalı demektir.

• IREQ dış cihazdan gelen interrupt istek işaretidir. Belirli donanımdan geçerek INT işaretine dönüşür ve analiz cihazında son bulur.



- IACK istekte bulunan cihaza isteğinin kabul edildiğini belirten işaretidir.
- CPU'ya gelmemesi gereken işareti ister CPU'da (maske yardımıyla), ister araya cihazında reddedebilmiz. Araya cihazında reddedersek bu istek CPU'ya ulaştırılmaz.
- IACK işaretini alan cihaz kim olduğunu, CPU'ya bildiren kimlik kodunu yayınlar.
- IACK alçağa çekilince IREQ işaretini istekte bulunan cihaz alçağa çeker, yani kendi isteğini ortadan kaldırır.

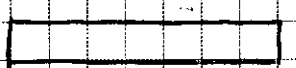
CPU'nun kesmeye cevabı

- Kesme isteği gelmeden önce kaşılmakta olan program ne olacak?
- Servis programının başlangıç adresini CPU nereden bulup program sayıcıya koyacak?

Yeni bir istek geldiğinde kaşılmakta olan programın kullanıldığı registerler stack adı verilen yerde saklanır. Bu stack ana bellekte olabileceği gibi CPU içinde de olabilir. Öncelikle PC (program sayıcı) içeriği saklanır.

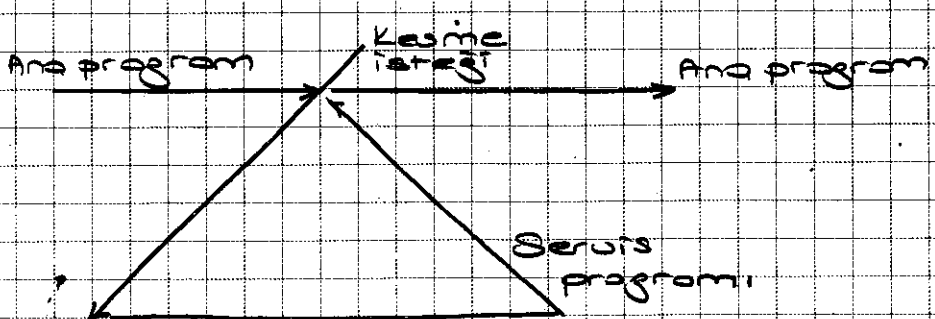
Az sayıda register içeren CPU'larda register-lerin içeriğinin saklanması otomatiktir, çok sa-yıda register içeren CPU'larda (Intel'in CPU'ları) bu işlem programcıya bırakılır. Bu register-lerin stacka itilmesi ve daha sonra ise çekilmesi istek emirlerinin başında ve sonun-da gerçekleştirilir. Stackın başlangıcı programcı tarafından belirlenir.

Stack belleği adresleyen registerler stack po-interdir.



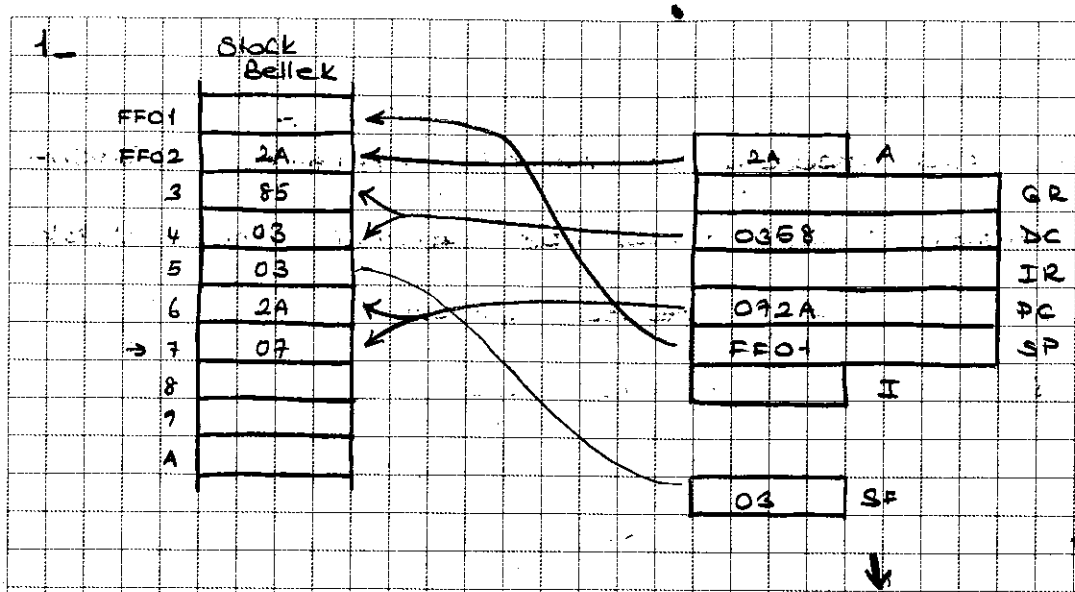
PUSH A } Stacka bilgi atar
PUSH BC

PULL A } Stackten bilgi çeker.

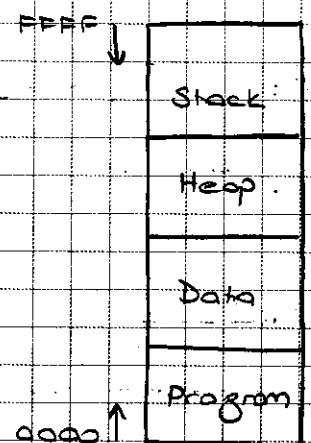


Ana program koşulurken kesme isteği gelduğinde servis programı koşarı tamamlandığın-da ana program kaldığı yerden koşmaya devam eder.

1-



Buradaki bilgileri stack belleğe topluz. Servis programı koşulduktan sonra bu bilgileri tekrar yerlerine yerleştiririz.



PULL emri önce stack pointeri bir artırır ve değerleri sırayla okur.

Servis programı koşulmadan önce kesilen programın register içerikleri stacka atılırlar. Bu işlemden önce interruptlar disable edilir. Registerler stacka eklendikten sonra RTI (Return to Interrupt) komutu ile interruptlar enable edilir. Sonra servis prog-

ramı koşulur.

2- CPU'ların çok sayıda interrupt girişinin olduğu varsayılmaktadır. Her interrupt girişi için CPU'nun baş vureceği fix bir adres vardır.

FFF8) INT   FFFC) NMI
FFF9)   FFFD)

3- Dış cihaz interrupt onaylama anında ya servis programının başlangıç adresini veya kendi (kimlik) kodunu CPU'ya iletir.

Servis programının başlangıç adresinin belirlenmesi bu 3 yöntemle yapılabilir.

* Base pointer register da kullanılmalı çünkü stacka ne kadar değer atılacağını bilmediğimizden tabanı ve tavanı tutmamız gerekiyor.

Kesme önceliği

1- Programlı öncelik belirleme: CPU interrupt gelecek tüm cihazları sırayla yoklar. Programlı G/Ç deki gibi boşa zaman harcamayız çünkü istek geldiği kesin, kimden geldiğine bakıyoruz.

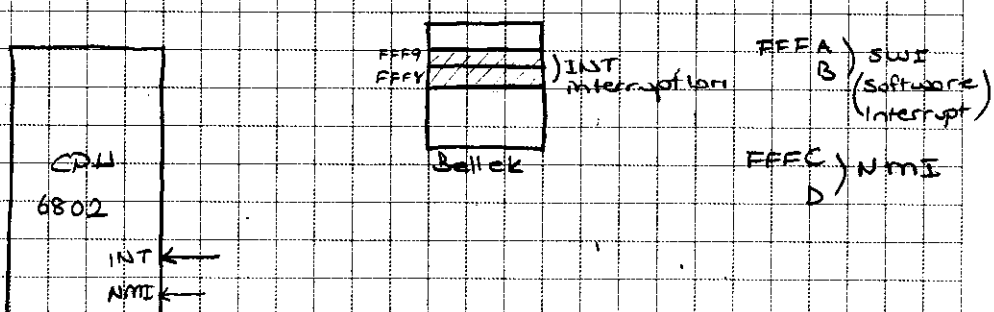
Öncelik sırasına göre istekleri yoklayan ve dış cihazın servis programının başlangıç adresini

hesaplayan bir programımız var.

- Intelde FFF8 - FFF9 'da servis programı olurduğu düşünülür. Motorolada ise bu adreste sadece servis programının başlangıç adresi vardır.
- En basit CPUda bile öncelikleri farklı 2 interrupt girişi vardır.

INT → Maskelenebilir interrupt girişi (öncelikli değil)

NMI → Maskelenemez (yüksek öncelikli) interrupt girişi



İstek geldiğinde kimden geldiğini ve servis programının adresinin nerede olduğunu bulmamız gerekiyor. CPU FFF8-FFF9 adreslerindeki içeriği aldı ve bunu önceliği belirleyen, kimden isteğin geldiğini belirleyen ve o isteğe bağlı olan servis programının başlangıç adresinin hesaplanmasını sağlayan program olarak değerlendirir.

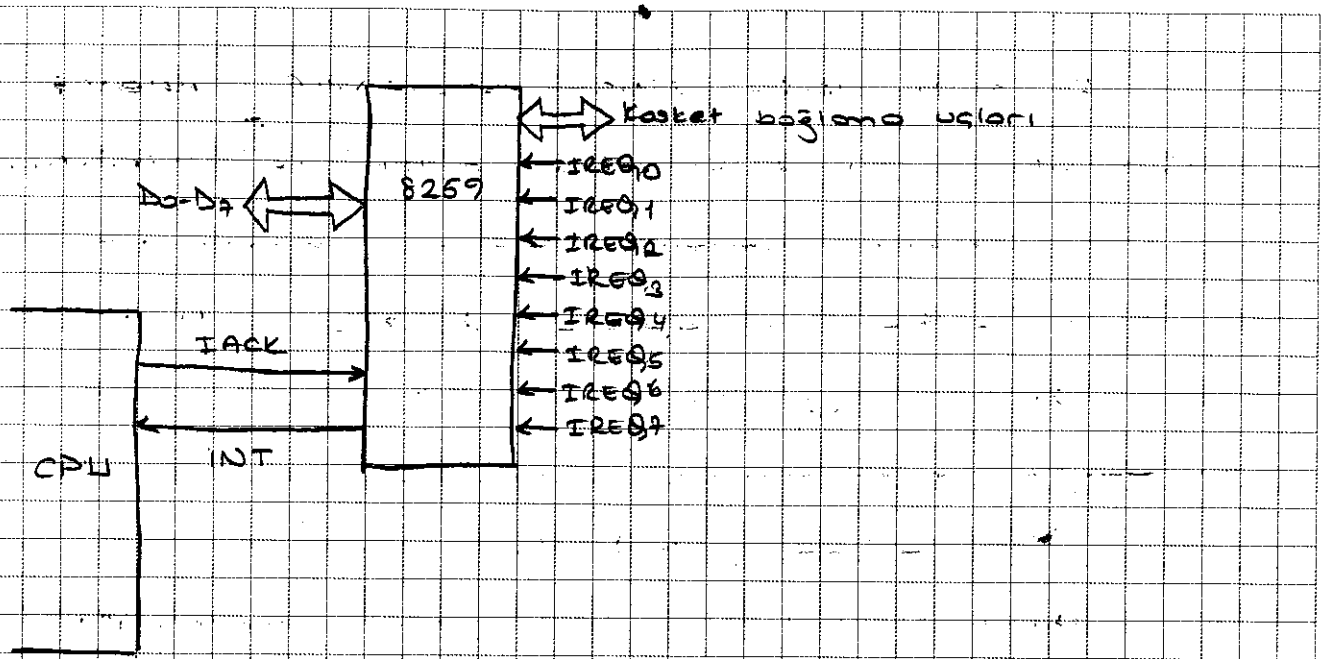
NWI'nın isteği INT'ten daha önceliklidir. NWI maskelenemez. Software ile NWI kontrol edilemez ve

bir kere NMI işareti CPU'ya ulaşırsa NMI, CPU'yu ele geçirir ve isteğini ne olursa olsun yaptırır. Bunu engellemek için CPU dışına özel bir donanım ekleyerek NMI'yi maskelenebilir hale getiriyoruz.

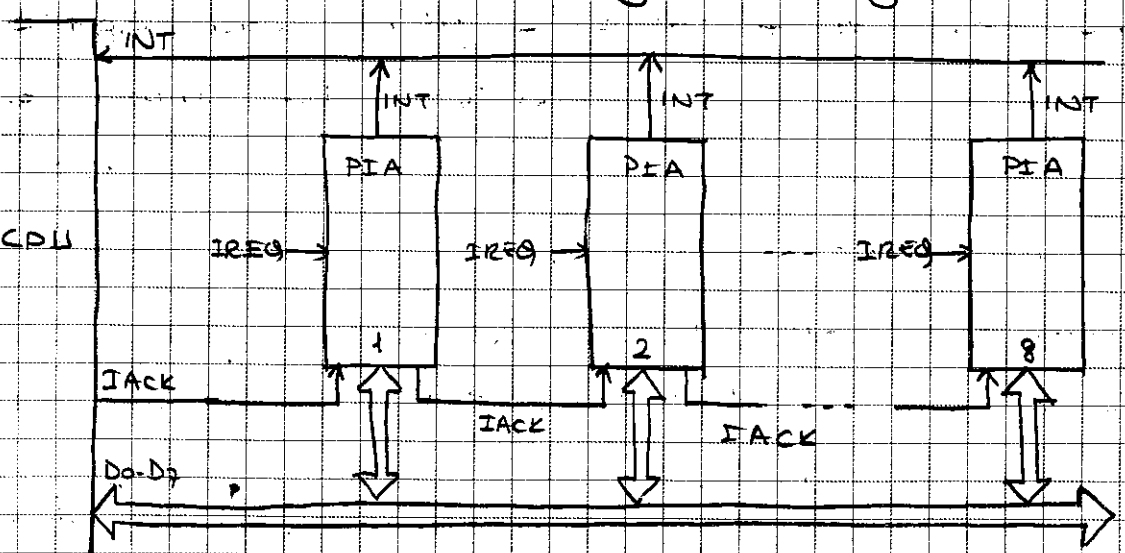
2- Öncelikleri farklı çok sayıda interrupt girişi olan CPU kullanmak: CPU'ya farklı öncelikli interrupt girişleri bağlayarak isteğin önceliğine hangi interrupt uçundan geldiğine bakarak karar veririz. Mesela 8086 CPU'sunda farklı öncelikli 6 interrupt girişi vardır. Her istek için ayrı adres olduğundan adres haplamaya gerek kalmaz.

6802 CPU'sunda da INT ve NMI uçları bulunur.

3- PIC (Programlı Interrupt Controller) kullanmak: Bu chip 8 tane interrupt ucuna sahiptir. 8259 olarak bilinir. Bu cihaz hem öncelik hem kimlik sorununu çözer. 1 veya 1'den fazla istek olduğunda CPU'ya INT işareti gönderir, CPU INACK işaretini PIC'a gönderir.



4- Poolyo zincir yöntemi: Bu yöntemde önceliği değiştirme şansımız yok. Akıllı analiz yöntemleri gerektirir ve 280 CPU'sunda göre bir yöntemdir.

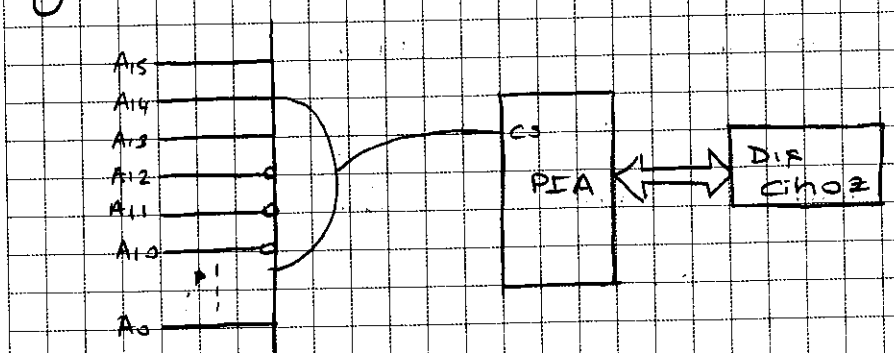


En yüksek öncelikli olan IACK ucu ilk olarak hangisine bağlanırsa odur. Burada 1. cihazın işi bitme

direkçe 2.'ye ve diğerlerine geçilemez. Öncelik değiştirmek için IACK bağlantısı değiştirilmelidir. Her cihazın işlerime sistem hazırlanırken (initialize sırasında) servis programının başlangıç adresi yazılır.

En büyük dezavantajı donanım değiştirilmeden öncelik sırası değiştirilemez. Bu da önceliği az olan cihazın işinin hiç yapılmamasına sürekli beklettilmesine sebep olabilir.

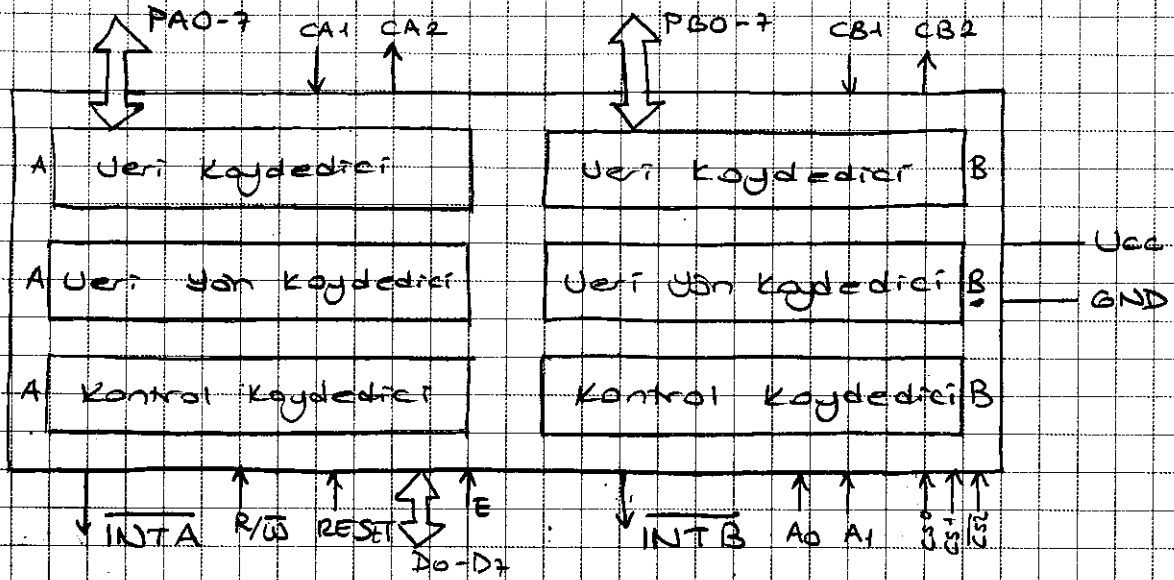
??? Plug and Play cihaz kendisini sisteme nasıl dahil eder ve bulur? - CPU ancak adresini bildiği cihaza erişir. CPU adres kod çözücüyle adresi üretir ve o adrese diş cihazı kodlayıcı.



Decoder (Adres 40xxx)

Paralel Giriş/Çıkış Arayüz Cihazı (6821 PIA)

Arayüzler genellikle 40 bacaklıdır. Bu da öyle.



Bu da INT çıkışı ya open collektörle bağlanarak ya da kapılanarak CPU'ya gider. Çünkü CPU'nun sadece 1 tane INT ucu var.

Adres bus'ın en anlamsız iki biti (A_0-A_1) de PIA'ya bağlanır. Bunlar A veya B taraflarını seçmeye yarar. Veri kaydedici ile veri yön kaydedicinin adresi ortak olduğundan PIA'de 4 adres vardır. A_0-A_1 sayesinde bu 4 adresten biri seçilir.