8085 A MI için anakart hafıza sistem tasarımı yapılacaktır. Buna göre Ep modüllerinden 1.si 0'dan 07FF'te bitmektedir, 2. Ep modülü ise 2000'den 27FF'te bitmektedir. RAM modülleri ise her biri 8k lık alan işgal etmektedir 1. RAM modülü 7000 H'ten, 2. RAM modülü C000 H'ten 3. RAM modülü E000 H'ten başlamaktadır. sistemi

a) sistemi 74LS138 kod çözücüsü, 6264 RAM tipi ve ihtiyacı karşılayan Epromları ile tasarlayınız.

b) sistene mümkünse fold back kullanarak tasarlayınız.

c) fold back kullanılması durumunda toplam kaç kB kaybı hesaplayınız.

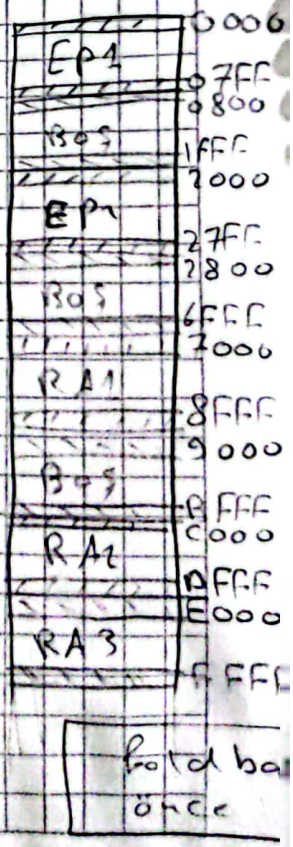
d) ileride sistene kaç adet RAM ve Ep eklenebilir (klasik sistem).

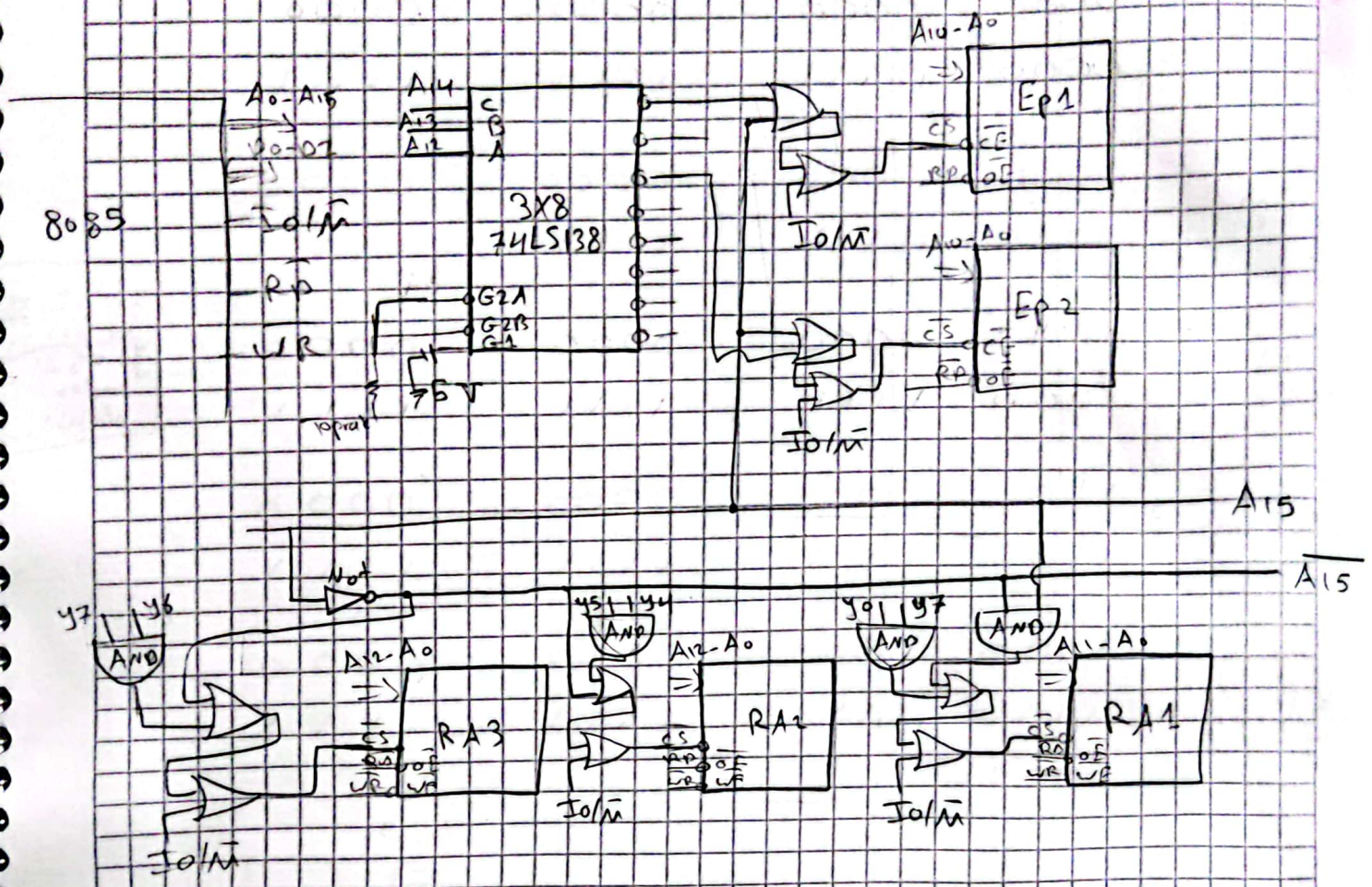
e) fold back ile tasarlandığında kaç adet RAM ve Ep eklenebilir.

Çözüm:

b)

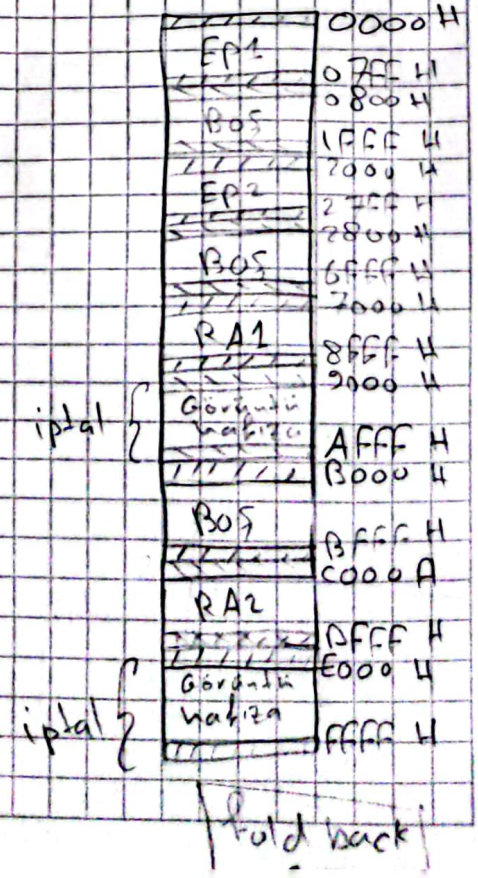
	A_5 A_4 A_3 A_2	A_1 (fold back)			
Ep1	0000	0000	0000	0000	0000
	0000	0111	1111	1111	07FF
Ep2	0010	0000	0000	0000	2000
	0010	0111	1111	1111	27FF
RA1	0111	0000	0000	0000	7000
	1000	1111	1111	1111	8FFF
RA2	1100	0000	0000	0000	C000
	1101	1111	1111	1111	DFFF
RA3	1110	0000	0000	0000	E000
	1111	1111	1111	1111	FFFF





c) • Toplam kayıb: 8k + 8k = 16 KB.

• 3 tane RAM yerine sadece 2 tane kullanabiliriz.

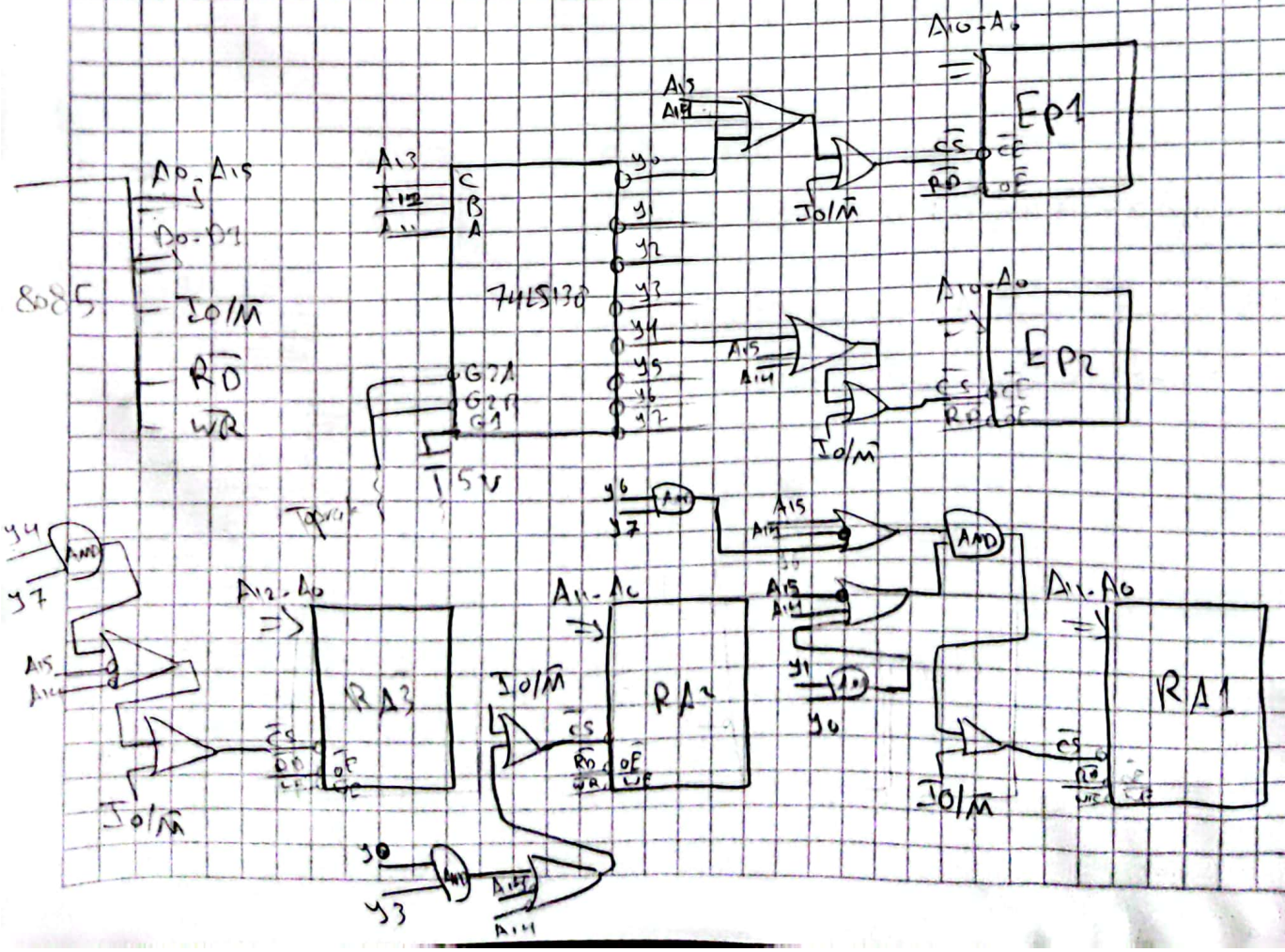


a)

	A_{15} A_{14} A_{13} A_{12}	A_{11}		A_0
Ep1	0000	0000	0000	0000
	0000	0111	1111	1111
Ep2	0010	0000	0000	0000
	0010	0111	1111	1111
RA1	0111	0000	0000	0000
	1000	1111	1111	1111
RA2	1100	0000	0000	0000
	1101	1111	1111	1111
RA3	1110	0000	0000	0000
	1111	1111	1111	1111

3 bit

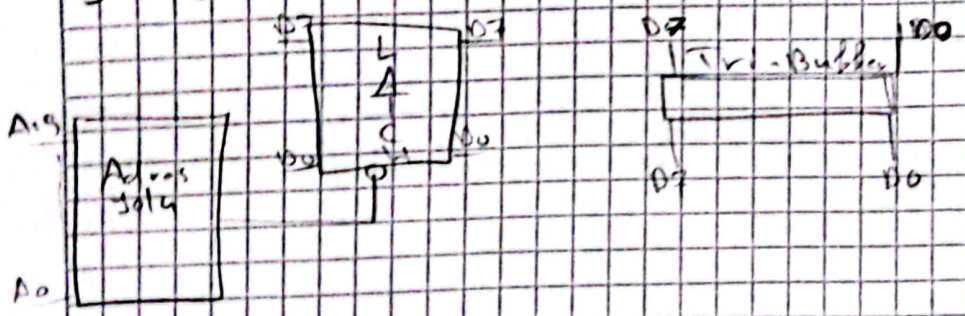
$y_6 \Rightarrow y_1$ kadar dahil



Hafta 7

- Giriş/Çıkış birimi tasarımları:

- G/Ç birimleri aynı hafıza birimleri gibi adreslenmektedir.
- Herbir birimin tetiklendiği adres veya adresler değeri bulunmaktadır.
- G/Ç birimlerini oluşturan asıl devre elemanları Tri-Buffer (tri-state) ve Latch'dir.
- Herbir aynı hafıza birimleri gibi _____ davranır. Gelen veri yüklenir ve yeni veri gelip giderek kadar saklanır. Yeni veri geldiğinde önceki veri silinir (Flip-Flop mantığı).



8085 MI sistemi için G/Ç birimi tasarlanacaktır. Buna göre tabloda gösterilen adres değerleri açan devreyi tasarlayınız.

a) Giriş birimi: F3H

Çıkış birimi: E00EH

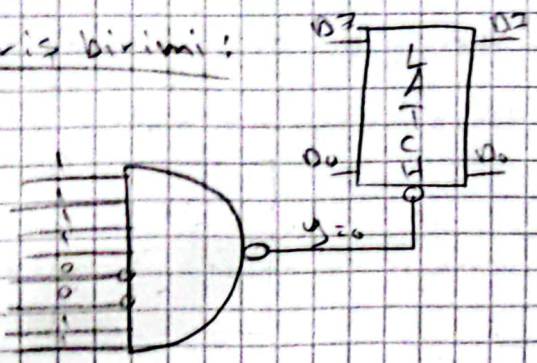
b)

	Giriş	Çıkış
Adres değerleri	FFH, F011H, F21	E000H - E00FH Bir adres - bir port

Not: Giriş birimi için Latch, çıkış birimi için Tri-Buffer.

Çözüm:

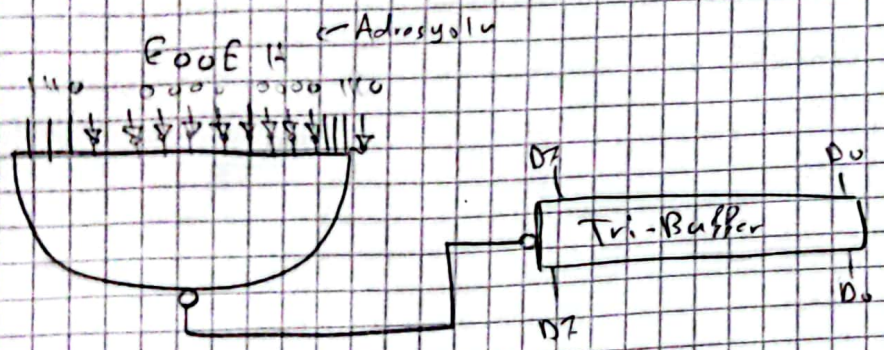
a) Giriş birimi:



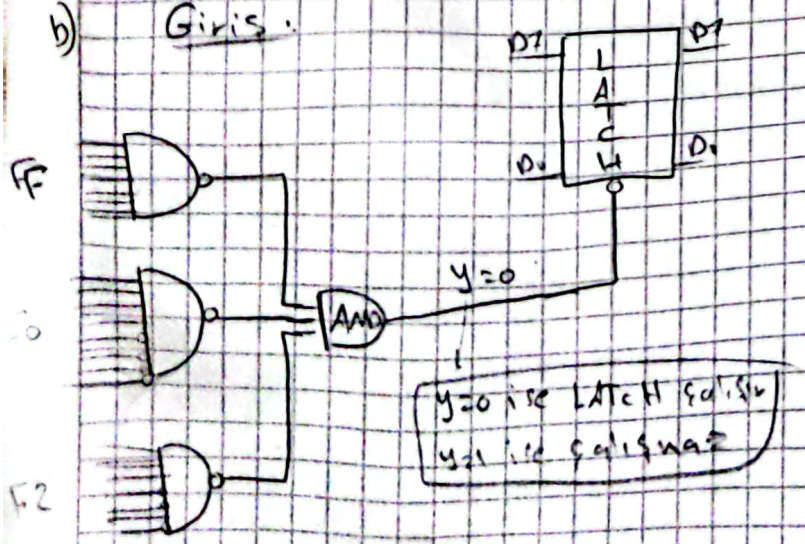
A_7 —	A_0	y
0 0 0 —	0	1
0 0 —	0 1	1
0 0 —	1 0	1
0 0 —	1 0 0	1
1 1 1 1	0 0 1 1	0

F3H

Çıkış



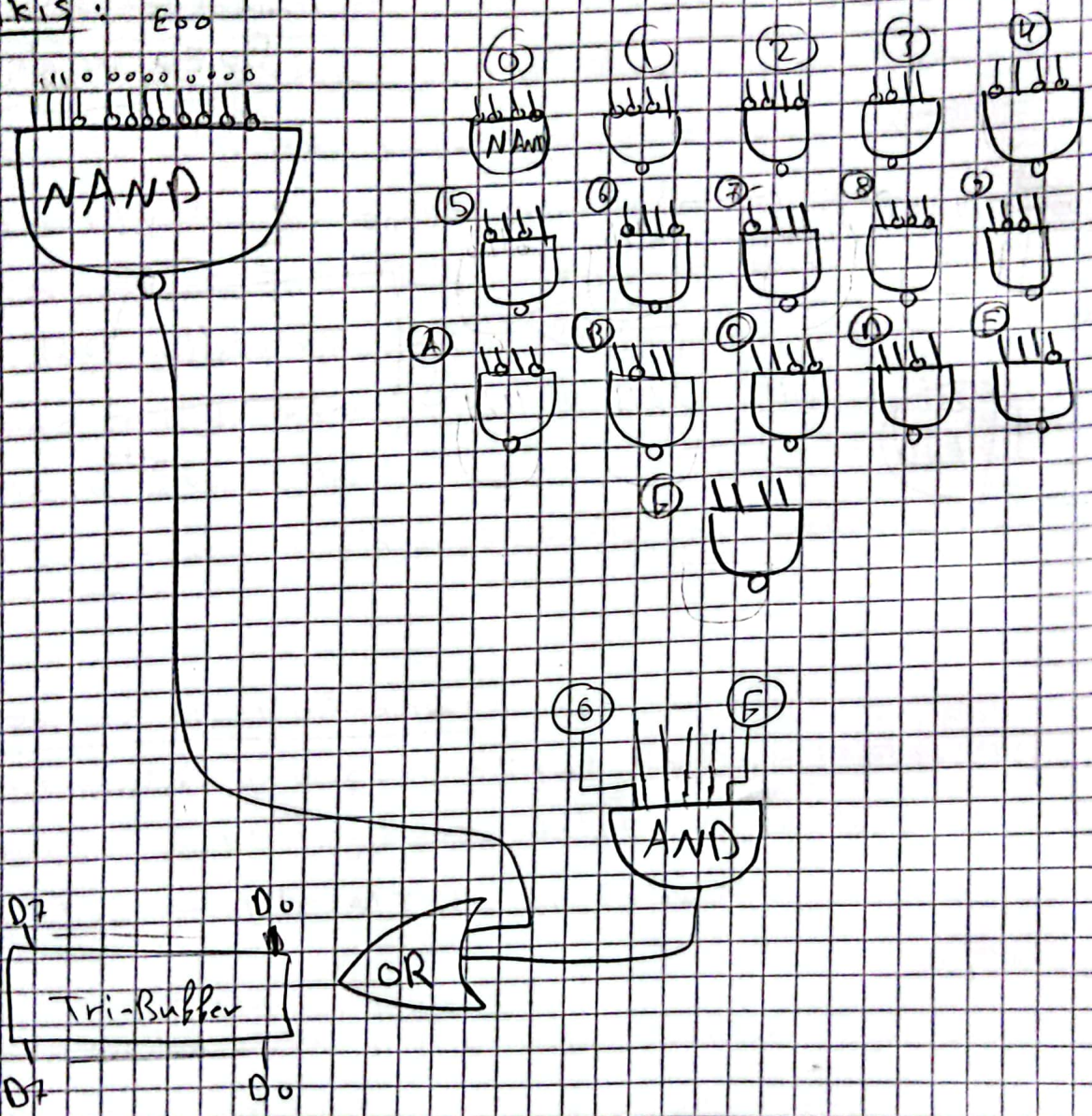
b) Giriş:



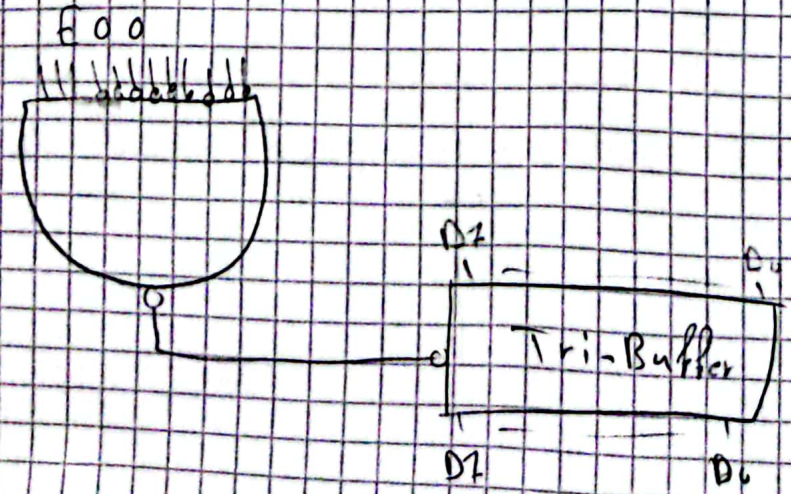
FF, F0, F2

1111 1111 1111 0000 1111 0010

Çıkış : E00

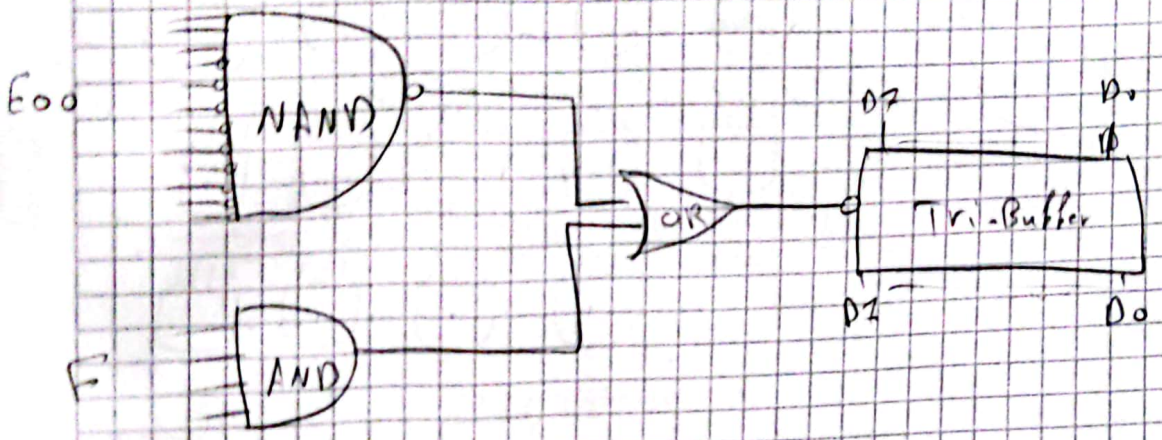


2. yol



E000 - E00E ise:

Bozucu devre



Hafta 9

- yazaç aktarımı:

Sayısal bir sistem sayısal donanım modüllerinin belli bir bilgi işlem olayını gerçekleştirmek üzere iç bağlantılardan bir araya gelmesinden oluşur. Bir sayısal bilgisayara aşağıdaki bilgileri verilmesiyle tanımlanır:

1) içerdiği yazaç kümesi ve bunların yapacağı işler.
2) yazaç içinde yazılı ikili bilgiler üzerine icra edilecek mikrobilgiler.
3) MI'in icrasını sağlayacak (başlatacak) denetim fonksiyonlar. mikro işlemler

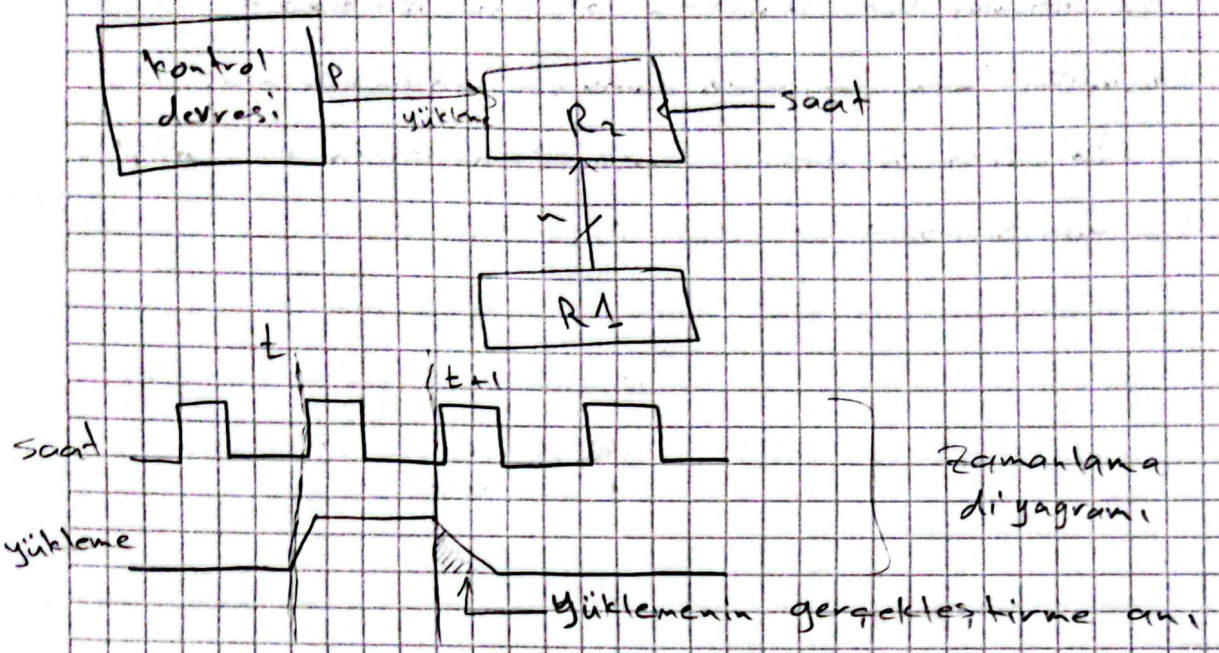
- yazaçlar arası aktarım:

Bilgisayardaki yazaçlar büyük harflerle gösterilir. yazaçları farklı boyutlarda olabilir, yazaçların bit genişliği genelde MI'in bit işleme kapasitesini gösterir.

- Bir yazaçtan ~~diğer~~ başka bir yazaca bilgi aktarımı sırasında aynı veri diğer yazaca aktarılır, sembolik olarak aşağıda gösterilir:

$R_2 \longleftarrow R_1$: R_1'den R_2'ye kopyalanır.

- Bu aktarma işlemi kontrol devresi tarafından kontrol edilir. Bunun blok şeması aşağıdaki gibidir:



- Blok şemasında görüldüğü gibi n bitlik veri R1 yazacından R2 yazmacına aktarılmaktadır ancak yükleme anı belli bir zaman aralığında gerçekleşmektedir. Kontrol devresi tarafından yükleme için açılan kapı saat darbesinin tetiklenmesiyle yükleme işlemi gerçekleşmektedir. Zamanlama şeması şekilde görüldüğü gibidir.

- Bu yazaçların birbirleriyle arasındaki veri transferi gerçekleştirmesi için yazaçların ortak veri yoluna bağlı olması gerekir. Her bir MI'in yazaçları belli bir teknikle bağlanır. Bu yazaçların ortak veri yolna bağlanabilmesi için aşağıdaki bağıntılar kullanılır.

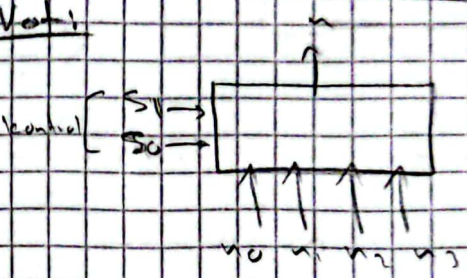
1) Yazaç sayısı = MUX boyutu
2) MUX sayısı = Yazmaç genişliği

② Toplam 4 adet her biri 4 bitten oluşan yazmaç bir MI tasarımı için ortak veriyoluna bağlanacaktır. Buna göre yazmaçların fonksiyon çizelgesi aşağıda verilmiştir. Sistemi bu fonksiyon çizelgesine göre tasarlayınız.

S_1	S_2	Seçile yazmaç
0	0	A
0	1	B
1	0	C
1	1	D

- 4 = Mux boyutu
  - ⇒ 4x1 Mux kullanılacaktır.
- Mux sayısı = 4

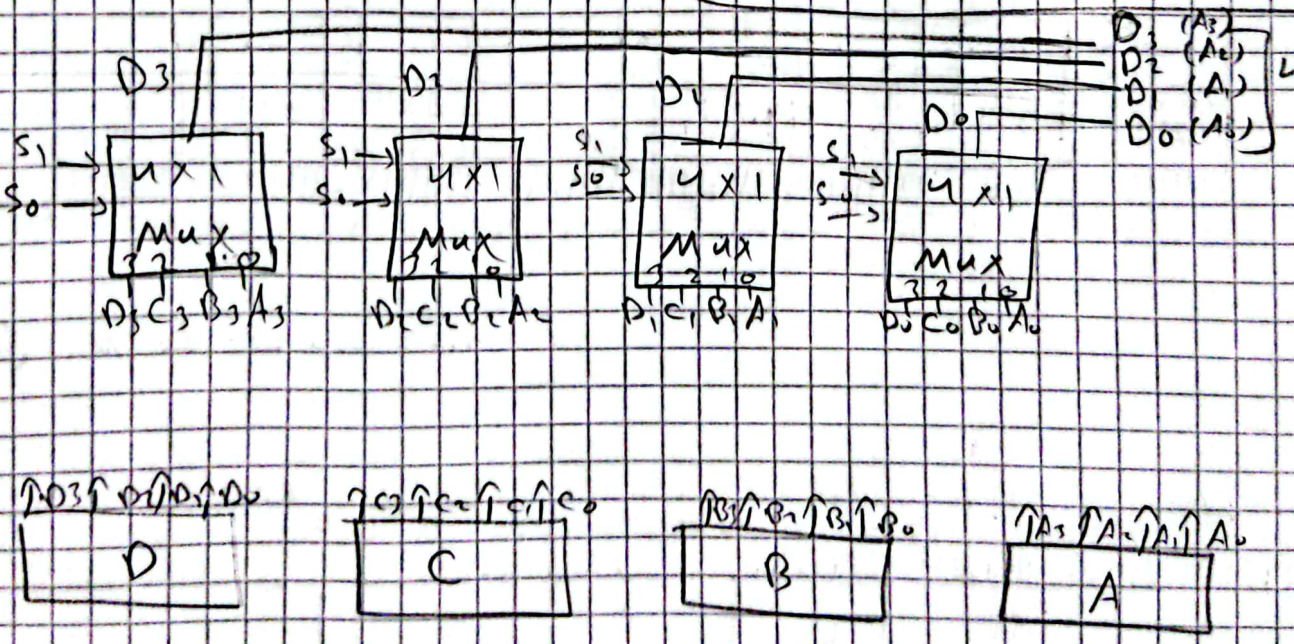
Not:



2^n = çıkış sayısı

kontrol ← k

2^k = giriş sayısı



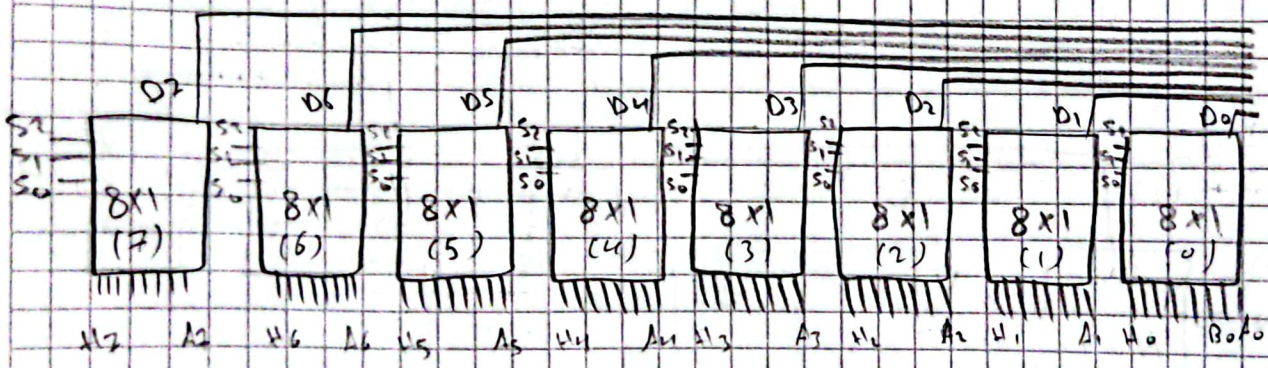
Hafta 10

Yazmaçların ortak veri yoluna bağlanması:

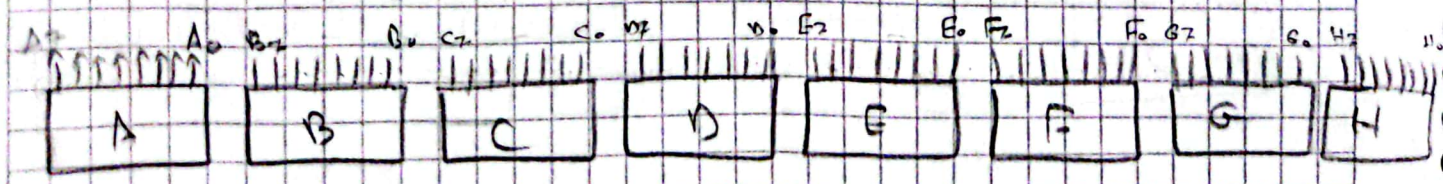
Her bir 8 bit olan toplam 8 adet yazmaç ortak veri yoluna bağlanacaktır. Sistemin fonksiyon çizelgesi aşağıda verilmiştir.

- yazmaç sayısı = 8 ⇒ Mux = 8 x 1
- Toplam Mux sayısı = 8

S_2	S_1	S_0	Seçilen yazmaç
0	0	0	A
0	0	1	B
0	1	0	C
0	1	1	D
1	0	0	E
1	0	1	F
1	1	0	G
1	1	1	H



بكل Mux يمكن اختيار السجل من 0 إلى 7.



Her bir 4 bit olan toplam 6 adet yazmaç ortak veriyolu bağlanacaktır. Sistemin blok. aşağıda verilmiştir.

S_2	S_1	S_0	Seçilen yazmaç
0	0	0	H
0	0	1	G
0	1	0	F
0	1	1	E
1	0	0	D
1	0	1	C
1	1	0	X
1	1	1	X

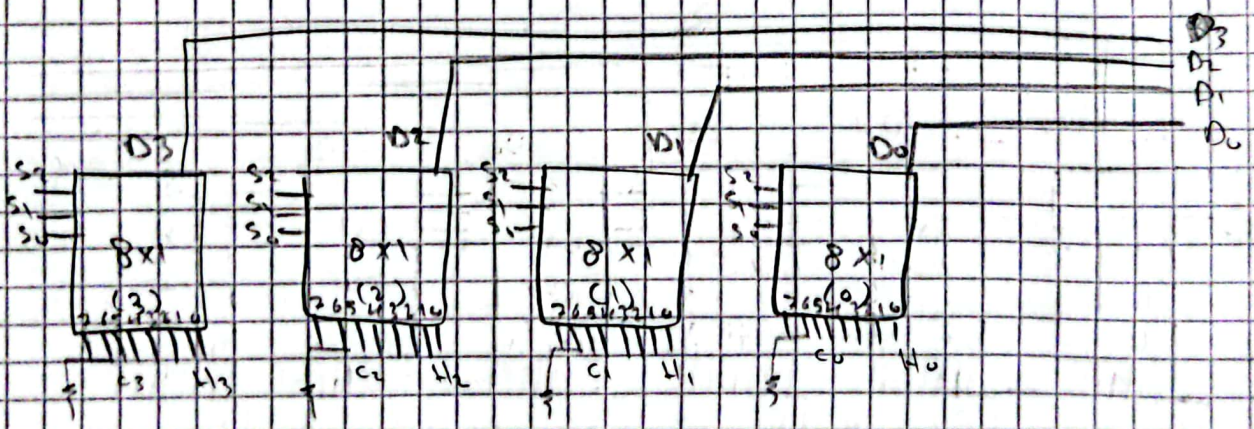
} çıkış yok

MUX boyutu = 8x1

n adet MUX

toplam MUX sayısı = 6

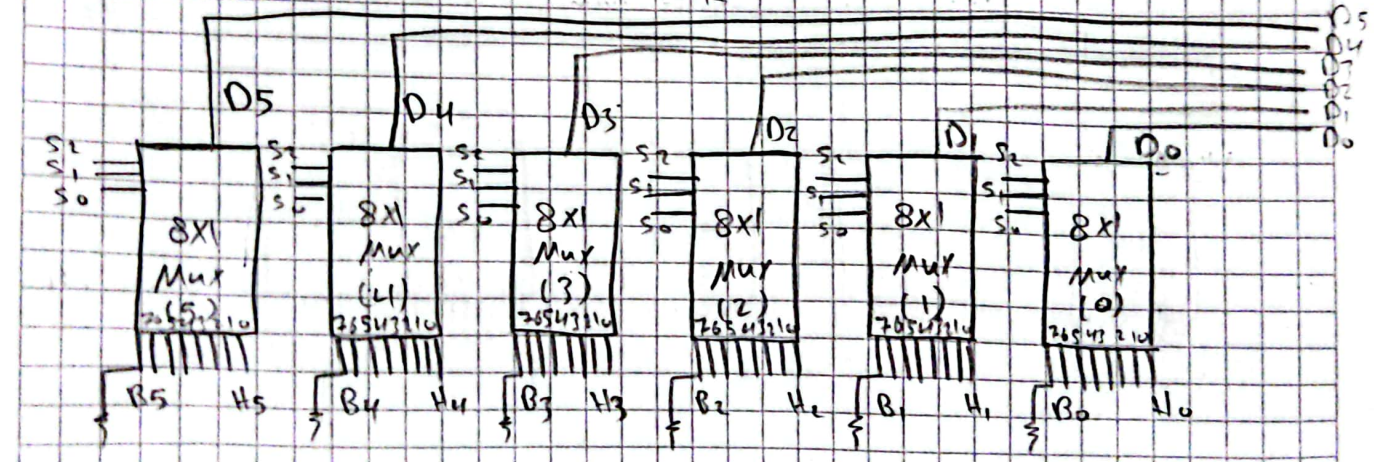
$2^n = 6 \Rightarrow n = 3$

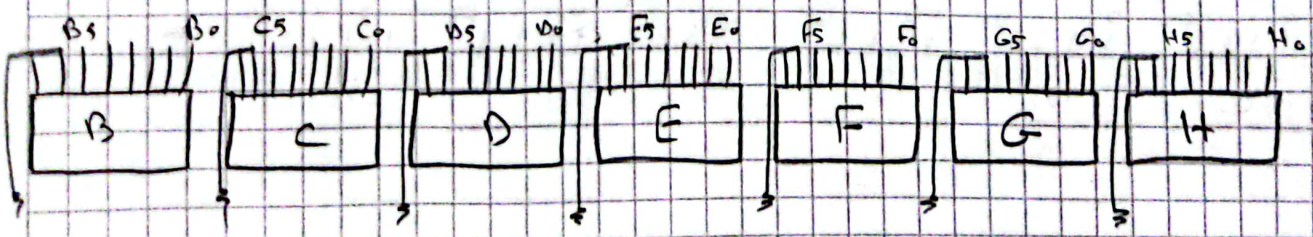


ödev

Her biri 6 bit olan toplam 7 adet yazmaç ortak veriyoluna bağlanacaktır. Sistemin blok. aşağıda verilmiş.

S_2	S_1	S_0	Seçilen yazmaç
0	0	0	H
0	0	1	G
0	1	0	F
0	1	1	E
1	0	0	D
1	0	1	C
1	1	0	B
1	1	1	X } çıkış yok





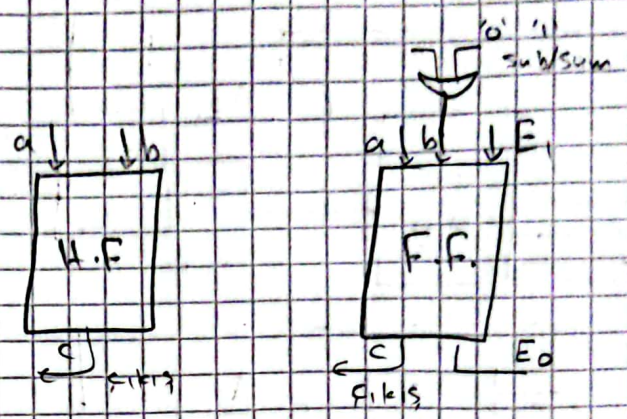
Toprak

Hafta 10

- MI'lerde AMB'lerin tasarımı:

- Hem çıkarma hem de toplama işlemi mantıksal kapılarla oluşturuluyordu.

- Toplama sadece:



- AMB (ALU) tasarımları:

- Aritmetik devreler:

⊛ Çıkarma, toplama gibi işlemler Mux ve FF'ler kullanılarak oluşturulur.

⊛ Aritmetik devrenin (yürüteceği işlem sayısı (fonk. sayısı))/2 = Mux. boyutu

⊛ Herbir Mux çıkışı ve sayı girişi FF'ler ile toplanır (Varsa elde girişleri de toplanır)

⊛ ALU tasarlanacaktır aritmetik devrenin fonk. çizelgesi aşağıda verilmiştir, tasarım 4-bitlik bir MI için yapılacaktır (4 bitlik aritmetik devreye ihtiyaç vardır)

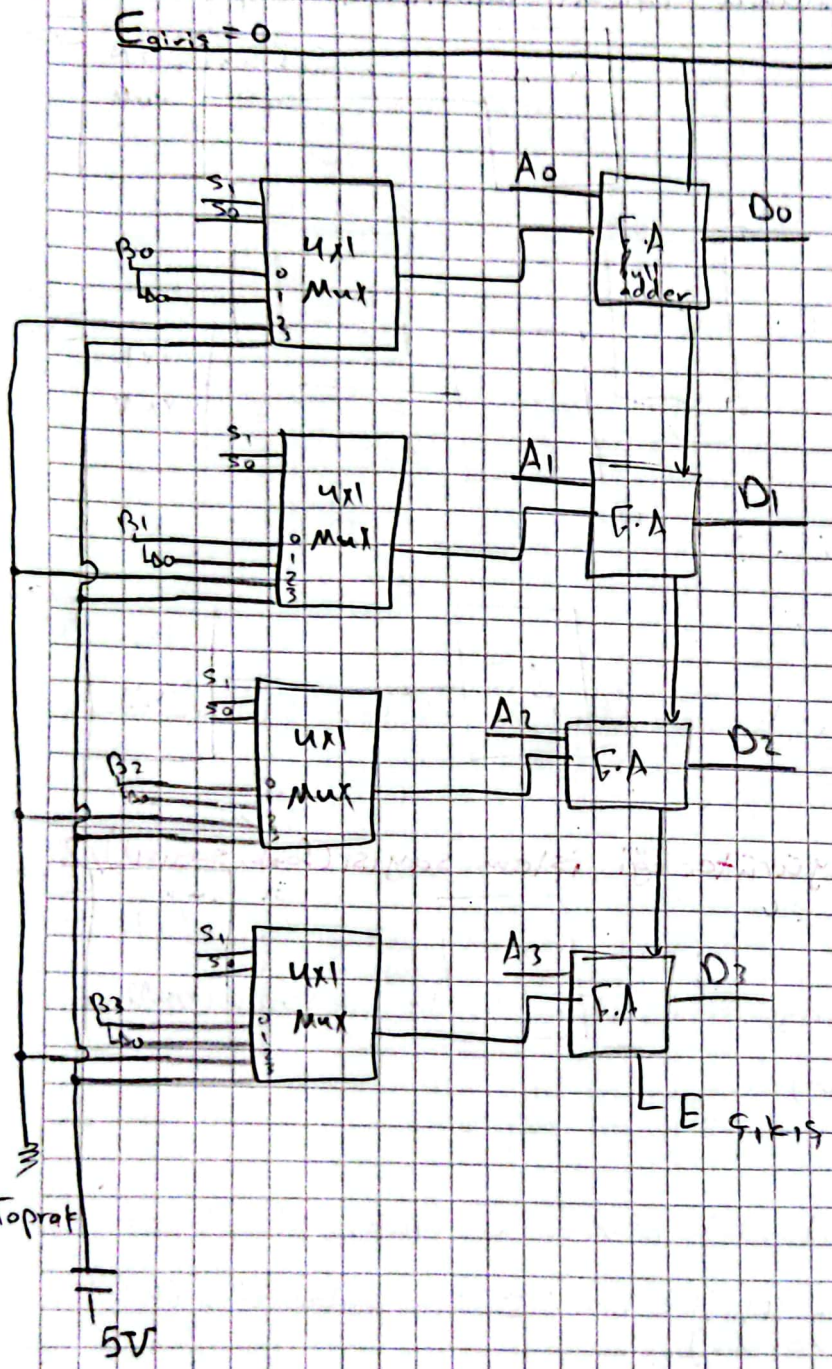
	S_1	S_0	E_i (elde)	Çıkış	Anlam
I	0	0	0	$D = A + B$	Toplama
	0	0	1	$D = A + B + 1$	Adet ile toplama
II	0	1	0	$D = A + \overline{B}$	Borç ile çıkarma
	0	1	1	$D = A + \overline{B} + 1$	Çıkarma
III	1	0	0	$D = A$	A'nın aktarımı
	1	0	1	$D = A + 1$	A'nın değerini '1' arttırma
IV	1	1	0	$D = A - 1$	A'nın " " azaltma
	1	1	1	$D = A$	A'nın aktarımı

ortak

8 adet fonk.

4(x) adet Mux kullanılmaktadır.

4-bit olduğundan dolayı toplam 4 adet Mux bağlanmalıdır.



$D = A - B = A + \bar{B} + 1$

$D = A - 1$ → 0
$= A + (\bar{1}) + 1$ → 5V

$D = A$
$= (A - 1) + 1$ → 5V

çıkacak bit ele alınıyor

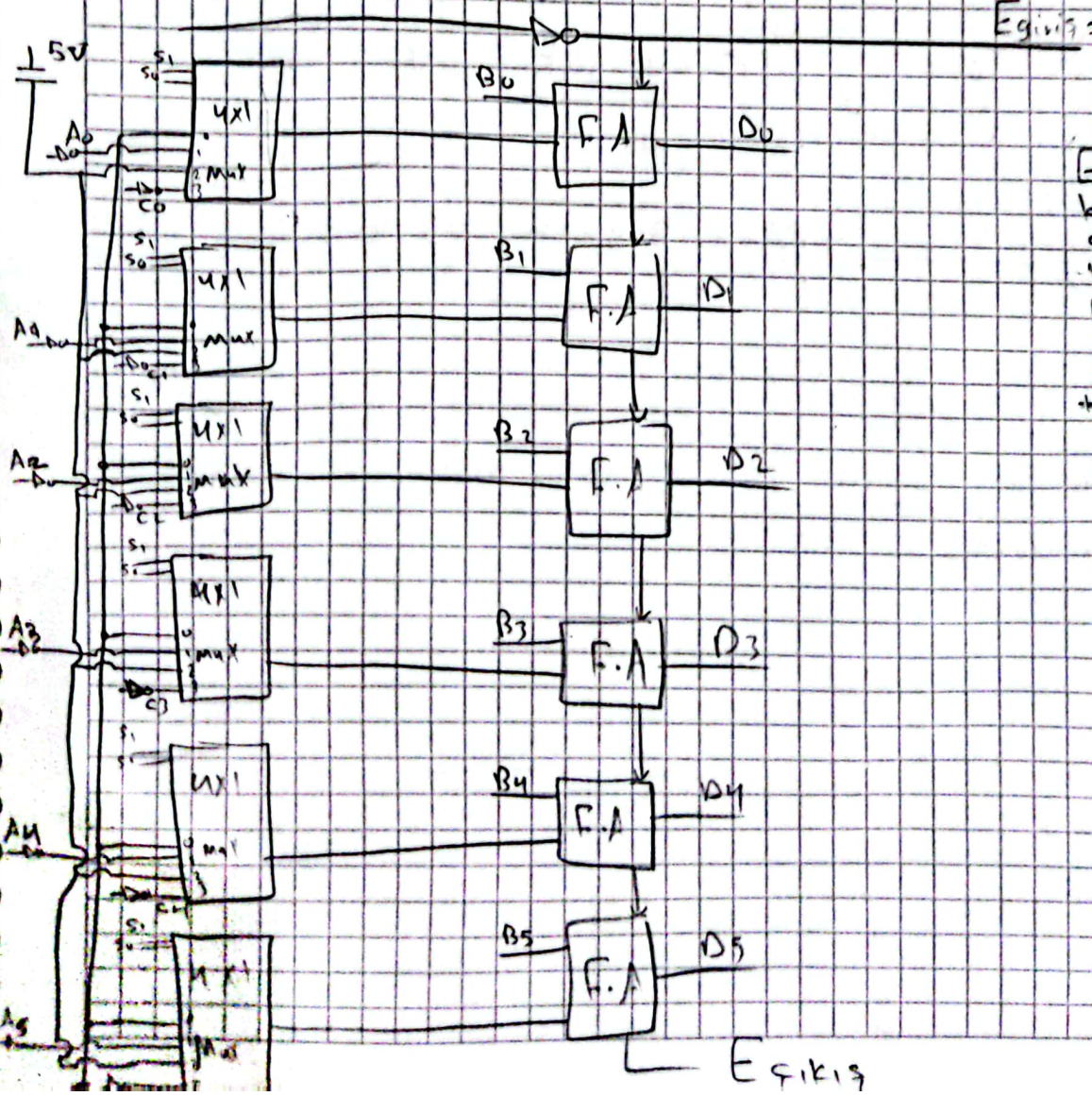
$1 - 1$
$= 1 + (\bar{1}) + 1$
$= 1 + 0 + 1$
$= 10$ → Sonucumuz

ele gidiyor

6 bitlik MI için ALU tasarlanacaktır:

	S_1	S_0	E	Çıkış
I	0	0	0	$D = B + 1$
	0	0	1	$D = B$
II	0	1	0	$D = B + \bar{A} + 1$
	0	1	1	$D = B + \bar{A}$
III	1	0	0	$D = B \Rightarrow D = (B-1) + 1$
	1	0	1	$D = B - 1$
IV	1	1	0	$D = B + \bar{E} + 1$
	1	1	1	$D = B + \bar{E}$

ortak olduğundan F.A.'ya bağlanır



E'de Not kullandık çünkü hep E=0 ise çıkışta +1 var

Hafta 11

(*) Aşağıda fonk. çizelgesi verilen bir MI tasarımı için AMB oluşturulacaktır. Sistemi FA ve MuX kullanarak gerçekleştiriniz. (Not: 6 bitlik işlemi için)

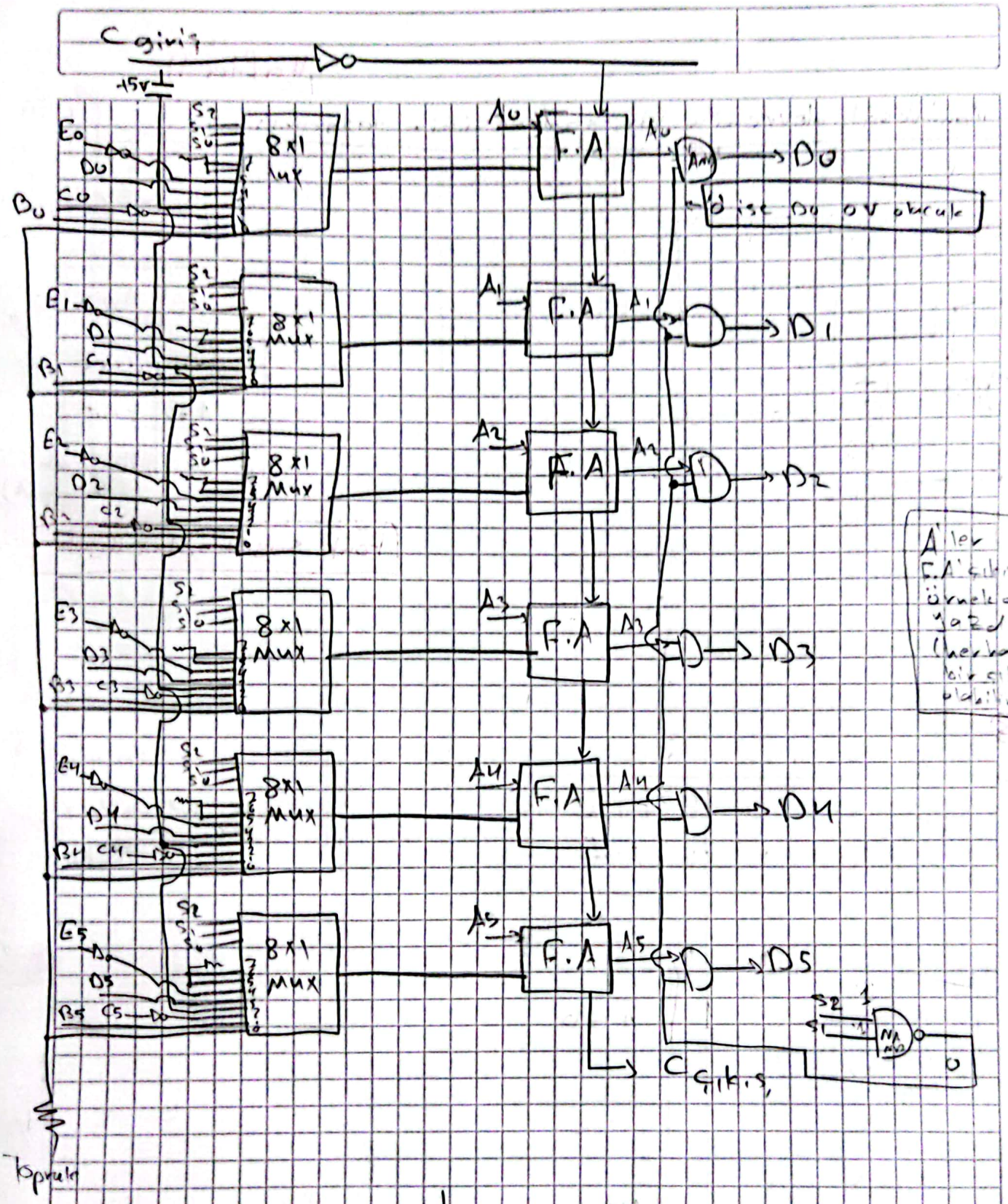
	S_2	S_1	S_0	$C_{giriş}$	işlem
I	0	0	0	0	$A + \boxed{1}$
II	0	0	0	1	A
III	0	0	1	0	$A + B + \boxed{1}$
IV	0	0	1	1	$A + B$
V	0	1	0	0	$A + \overline{C} + \boxed{1}$
VI	0	1	0	1	$A + \overline{C}$
VII	0	1	1	0	A
VIII	0	1	1	1	$A - 1$
	1	0	0	0	$A + D + \boxed{1}$
	1	0	0	1	$A + D$
	1	0	1	0	$A + \overline{E} + \boxed{1}$
	1	0	1	1	$A + \overline{E}$
	1	1	0	0	Çıkış yok OV
	1	1	0	1	
	1	1	1	0	
	1	1	1	1	

$(A-1) + \boxed{1}$
$(A + \overline{1} + 1)$

$C_{giriş} = 0$ iken çıkışta +1 var ondan $C_{giriş}$ tersleyici bağladık

$$\frac{\text{Toplam fonk. sayısı}}{2} = \text{MuX Boyutu} \Rightarrow \frac{16}{2} = 8 \quad \boxed{8 \to 1 \text{ MuX}}$$

S_2 ve S_1 1 ise çıkış yok, ondan ona göre ayarlarız.



Ailer F.A'ya girişinde örnek olarak yazıldı (herhangi bir giriş olabilir)

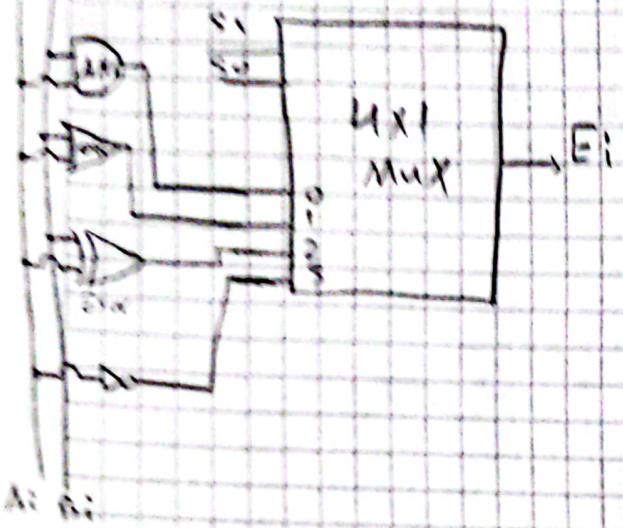
S_2	S_1	A_0	O
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

NAND ve AND'den sonra

6 ve 7 girişleri toprağa

Hafta 11

- Mantıksal devrenin bir kademesinin tasarımı:



fonk. çizelgesi

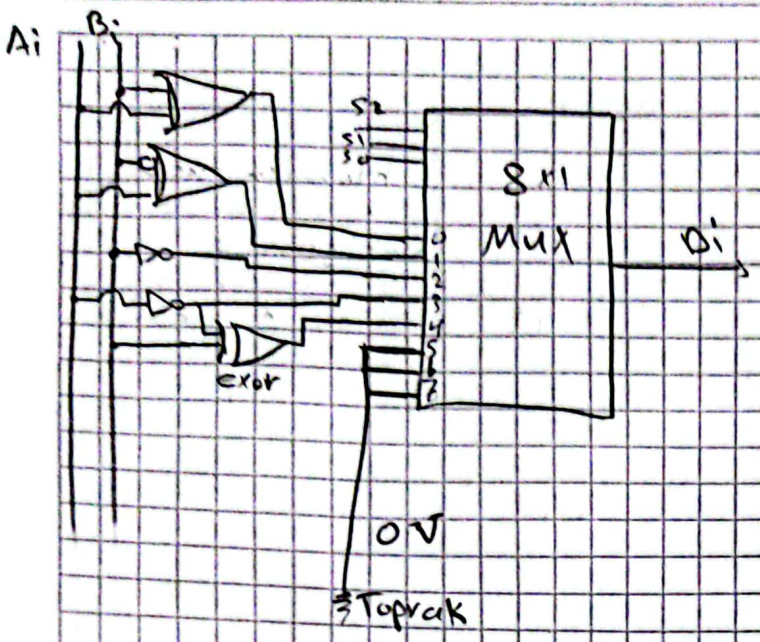
S_1	S_0	çıkış	işlem
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	Exor
1	1	$E = \bar{A}$	Tümleyen (complement)

$\boxed{\text{Fonk. sayısı = MuX Boyutu}}$

(*)

S_2	S_1	S_0	çıkış
0	0	0	$E = A \vee B$
0	0	1	$E = A \vee \bar{B}$
0	1	0	$E = \bar{B}$
0	1	1	$E = \bar{A}$
1	0	0	$E = \bar{A} \oplus B$
1	0	1	çıkış yok
1	1	0	
1	1	1	

yukarda fonk çizelgesi verilen mantıksa devrenin tek kademe dijital devresini çiziniz



- <u>Kaydırma devreleri:</u>

yazmaç içinde bulunan bir değerin bir bit sağa yada sola kaydırılması sağlar. Kaydırma devreleri için standart olarak yazmaç genişliğinden bağımsız olarak 2x1 Mux kullanılır. Ayrıca yazmaç genişliği kadar Mux kullanılması gerekir

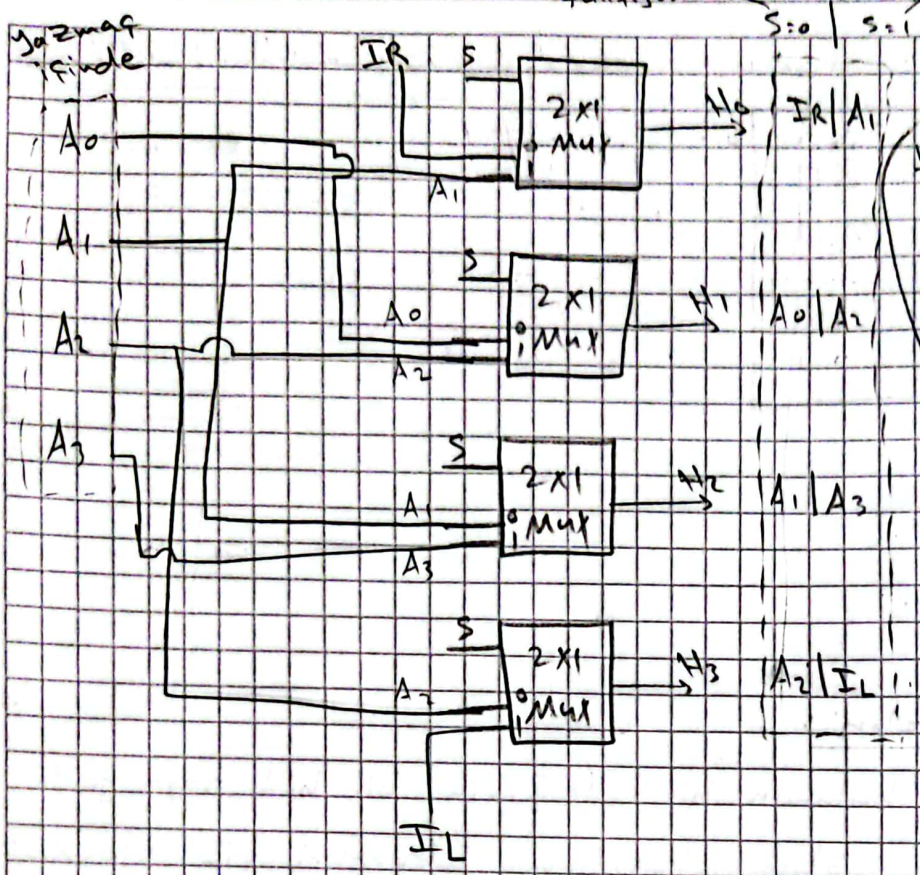
Seçilmiş	Çıkış H_0	H_1	H_2	H_3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

4 adet 2x1 Mux

ortakları önce bağlıyoruz (daha iyi)

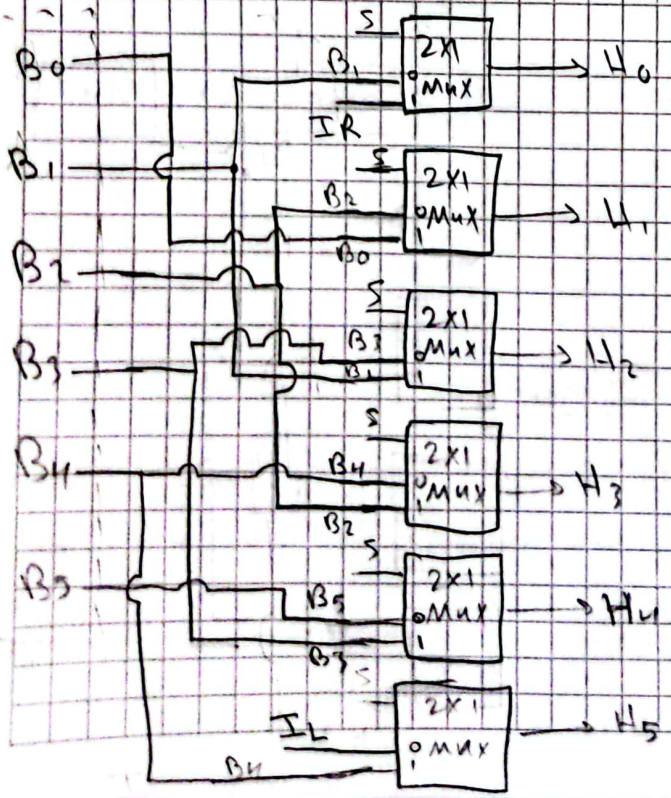
Muxta '0' girişi çalışıyor

Muxta '1' girişi çalışıyor



Tablodan her A nereye bağlı bakıyoruz (0 iken hangi H'e, 1 iken hangi H'e)

Seçilmiş	Çıkış H_0	H_1	H_2	H_3	H_4	H_5
1	IR	B_0	B_1	B_2	B_3	B_4
0	B_1	B_2	B_3	B_4	B_5	IL



Bunun sorusu

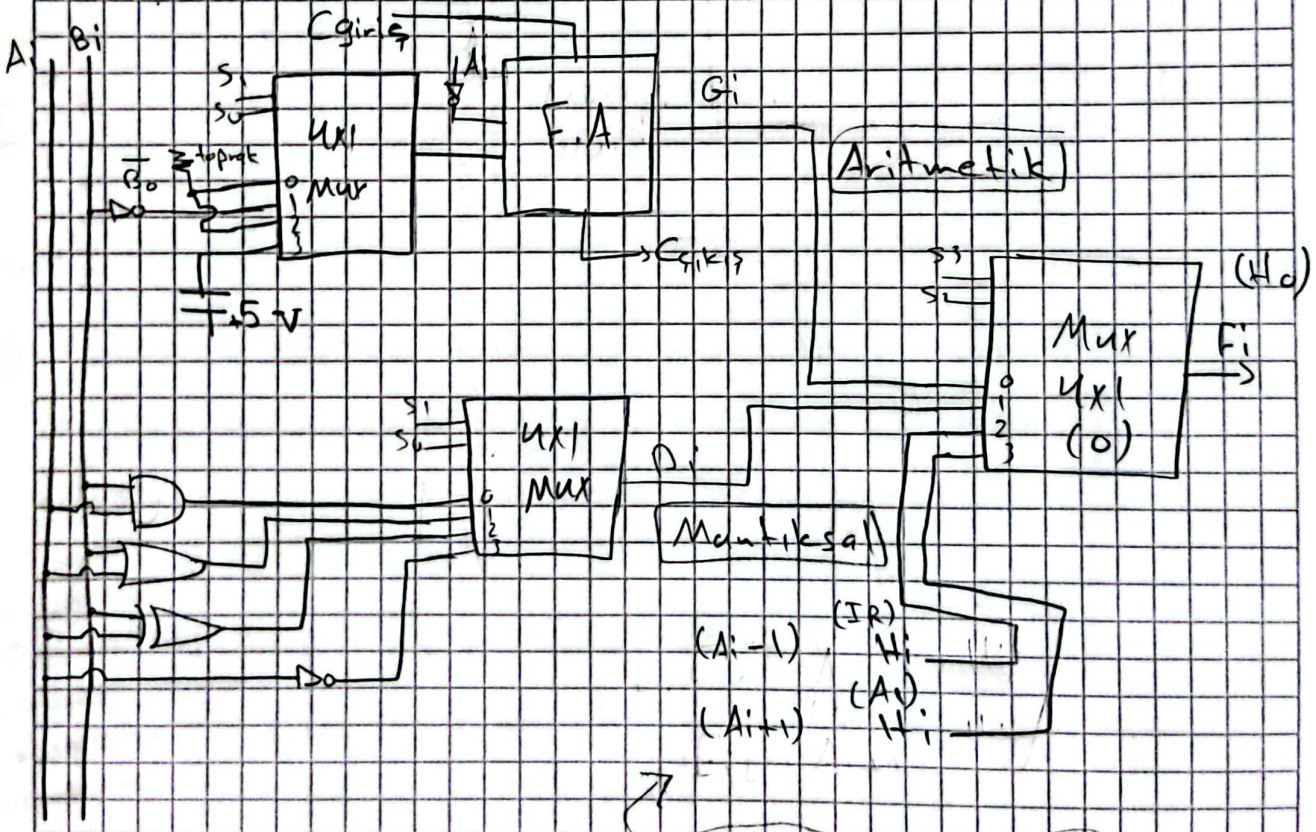
Bir MI tasarlanacaktır. Buna göre AMB'nin sağ ve sola kaydırma işlemlerin kaydırma bölümün fonk tablosu yukarda verilmiştir. sistem 2*1 Muxlar kullanarak ve 6 bitlik işlemci kullanarak tasarlayınız.

Hafta 12

Bir AMB'nin fonk çizelgesi aşağıda verilmiştir. Sistemin blok şemasını çiziniz. (4 bitlik)

S_3	S_2	S_1	S_0	Cgiriş	İşlem	
0	0	0	0	0	$F = \overline{A}$	
0	0	0	0	1	$F = \overline{A} + 1$	
0	0	0	1	0	$F = \overline{A} + \overline{B}$	
0	0	0	1	1	$F = \overline{A} + \overline{B} + 1$	Aritmetik
0	0	1	0	0	$F = A$	
0	0	1	0	1	$F = A + 1$	
0	0	1	1	0	$F = A - 1$	
0	0	1	1	1	$F = A$	
0	1	0	0	X	$F = A \wedge B$	
0	1	0	1	X	$F = A \vee B$	Mantıksal
0	1	1	0	X	$F = A \oplus B$	
0	1	1	1	X	$F = \overline{A}$	
1	0	X	X	X	$F = SHR\ A$	kaydırma
1	1	X	X	X	$F = SHL\ A$	

Bunlar Cgirişleri geliyor, 0 ise +0, 1 ise +1



S_3	S_2	
0	0	Aritmetik
0	1	Mantıksal
1	0	SHR
1	1	SHL

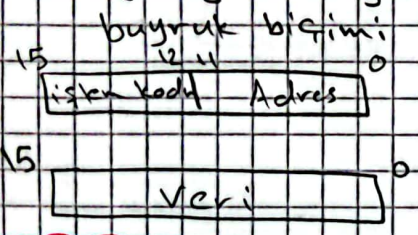
A_0 ise SHR çıkışı A_1 (Ai+1), SHL çıkışı IR (Ai-1)

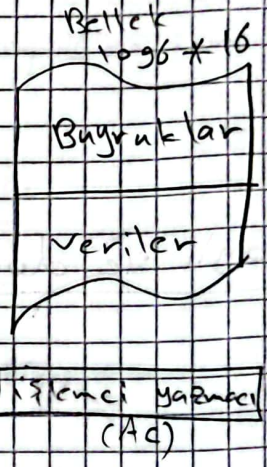
A_1 ise SHR çıkışı A_2 (Ai+1), SHL çıkışı A_0 (Ai-1)

Aynı anda 1 bit geliyor, Ai'de

- **Buyruk kodları ve buyruk yapıları:**

Bir MİŞ tasarımı yapılmadan önce o işlemcinin icra etmesi istenen işlemler belirlenir, bu işlemlerin uzunluğun doğrudan buyruk kodu genişliğini etkiler, Buyruk kodları işlem kodu bölümü ve adres bölümü olarak 2 parçadan oluşur. İşlem kodumun değeri işlem temsil eder, Adres değeri ise kullanılacak olan verinin adres değerlerini temsil eder saklanmış program yapısında aşağıda blok şemada görüldüğü gibi buyruklar ve veriler bellekte depolanır.



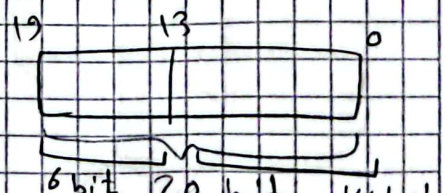


1 $2^{\text{işlem kodu}}$ = icra edilebileceğ edilebilecek işlem kapasitesi

2 $2^{\text{adres genişliği}}$ = Hafıza kapasitesi

(*) Toplam 16 kB'lık hafızayı sürebilen, toplam 20 bit'lik buyruk uzunluğuna sahip bir işlemcinin max icra edebileceği işlem kapasitesini hesaplayınız.

- $2^6 = 64$ farklı işlem icra edebilir.



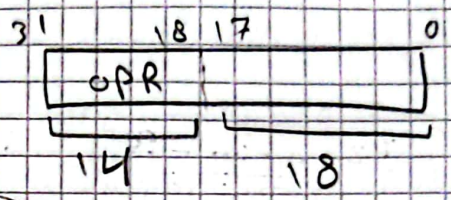
- $16\ kB \Rightarrow 2^4 * 2^{10} = 2^{14}$ (adres genişliği)

2^6

Belli bir ihtiyacı karşılamak için MI tasarlanacak. Buna göre tasarlanacak işlemcinin toplam 16384 (2^{14}) farklı işlem yerine getirmesi istenmektedir. İşlemcinin kapasitesi 32 bit olması istendiğine göre tasarlanacak bu işlemcinin sürebileceği max hafıza kapasitesini hesaplayınız.

$16384 = 2^{14}$

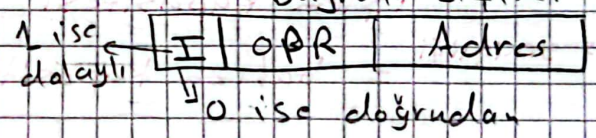
32 bit'lik MI

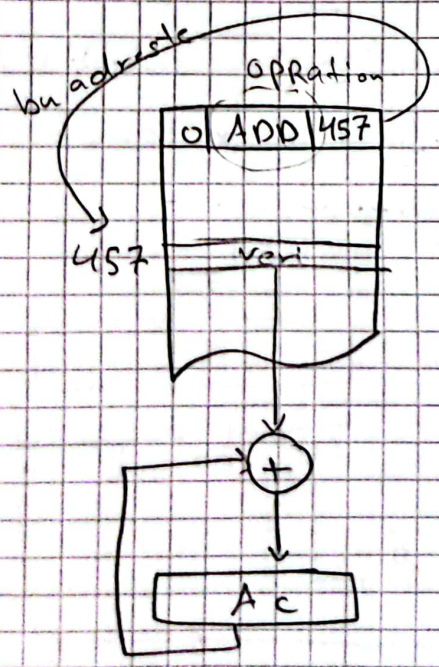


hafıza kapasitesi ← $2^{18} = 2^{8} * 2^{10} = 256\ kB$

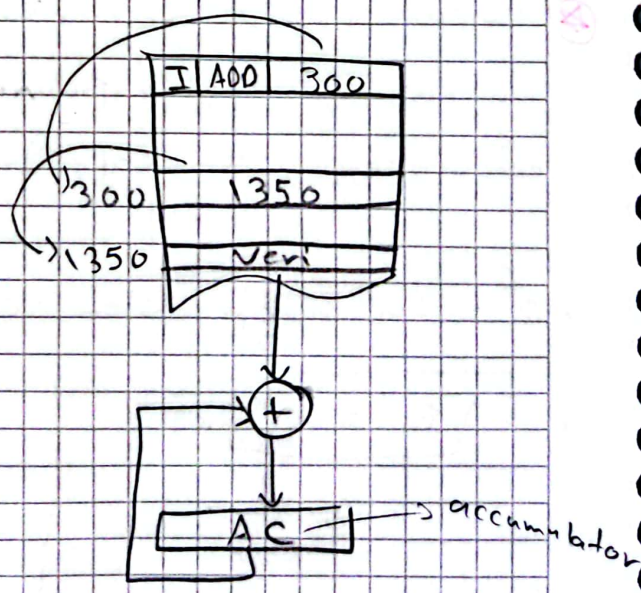
- Dolaylı ve doğrudan adresleme modu:

Buyruk biçimi





Doğrudan adresleme



Dolaylı adresleme

8192 farklı işlem yerine getirecek MI için buyruk kümesi tasarlanacaktır. Doğrudan ve dolaylı adresleme modları bulunacaktır. Buyruk genişliği 32 bit olduğuna göre:

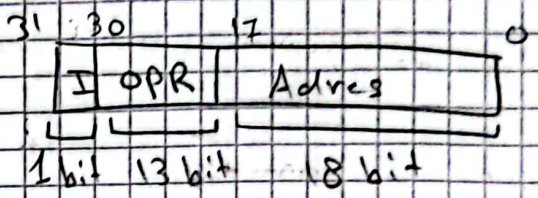
a) MI sürebileceği hafıza kapasitesi.

b) Buyruk kümesini tasarlayınız.

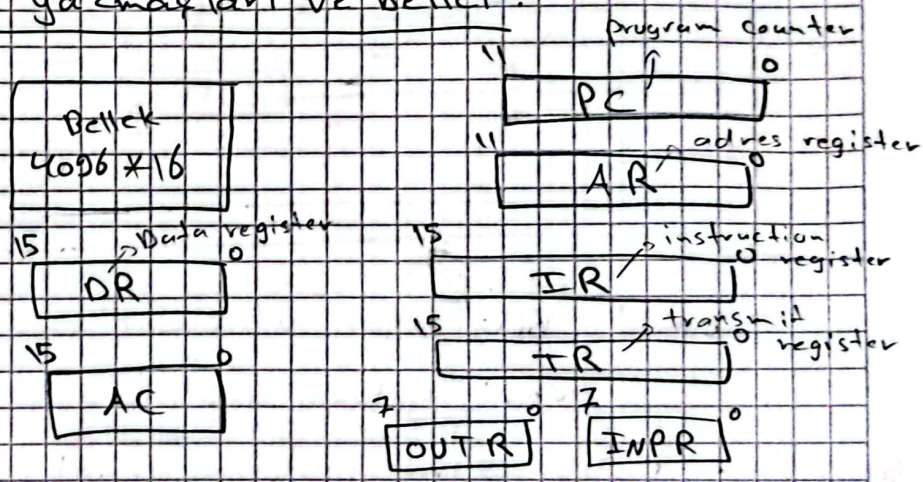
Çözüm:

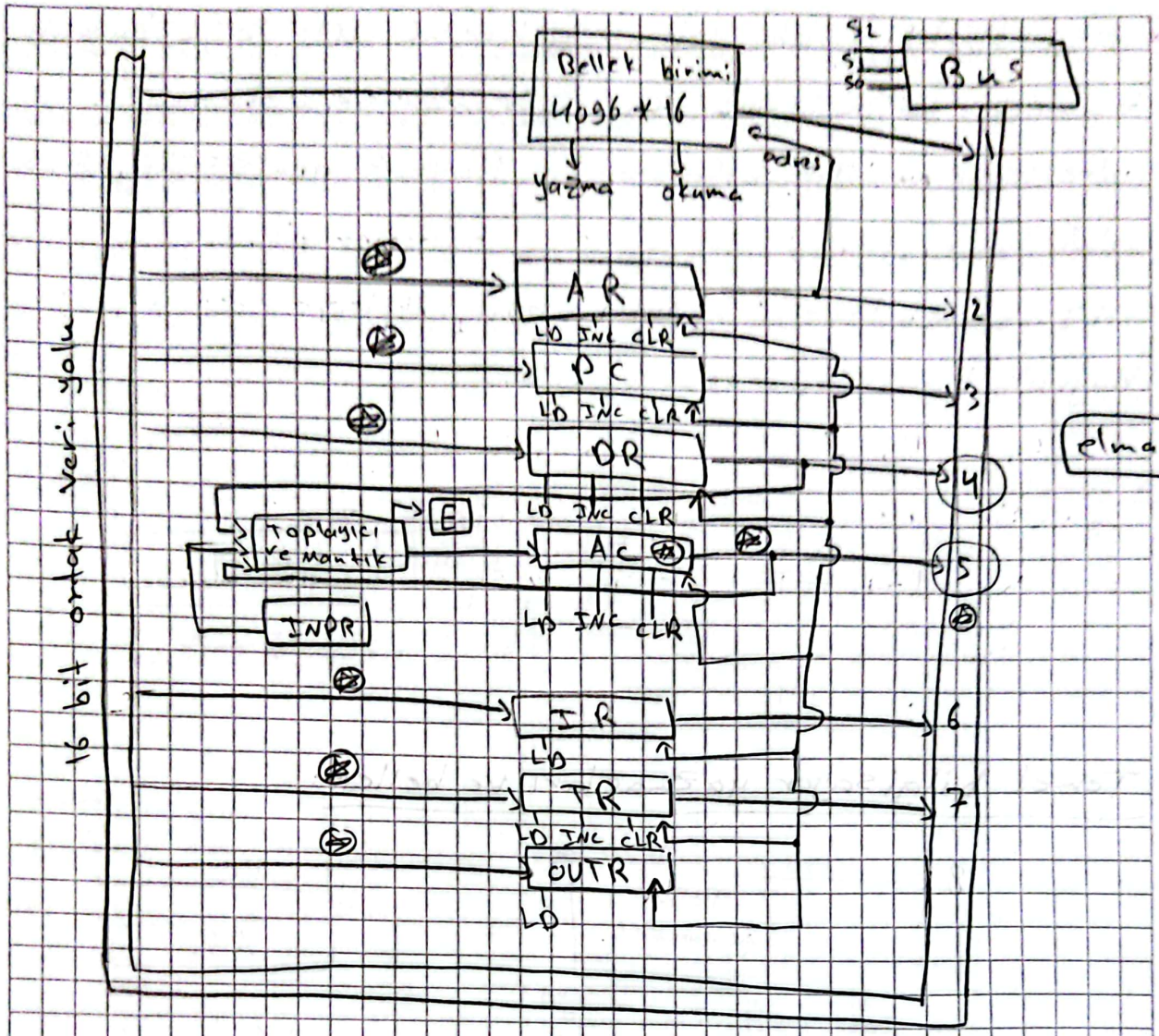
a) $2^{18} = 2^8 * 2^{10} = \underline{\underline{256\,kB}}$

b)



- Temel bilgisayar yazmaçları ve bellek:





Temel bilgisayar yazmaçların ortak veri yoluna bağlanması

☆ Sistem çalışmasını gösteren örnek:

DR ← AC — ①

AC ← DR — ②

Sadece DR'nin LD'i açılır, tüm yerlerde elma var ama sadece o çalışır

yukarıda verilen işlemlerin izlenmesi için icra edilmesi gereken adımlar.

① AC → ortak veri yoluna / $S_2\ S_1\ S_0 = 101$

⇒ DR → LD pin'i tetiklenir

② $S_2\ S_1\ S_0 = 100$ ⇒ AC → LD pin'i tetiklenir

ucunu (الشرط / المدخل LD)

- Buyruk süresi:

Bellekte bulunan bir program bir dizi buyruktan oluşur, her bir buyruk süreci bir takım alt parçalardan oluşur, bunlar alt süreçler veya süreçler denir.

Temel bilgisayarda her bir buyruk aşağıda süreçlerden oluşur:

1) FETCH (Bir buyruğun bellekten alınıp getirilmesi)
2) DECODE (buyruk kodunun çözümmesi)
3) Eğer bu adres dolaylı adresleme ise etkin adresin okunması.
4) Buyruğun icrası 4. adımın

örn: AL Getir kodunu çöz örneği:

T_0: $AR \leftarrow PC$

T_1: $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$ (M[]: Hafıza)

zaman

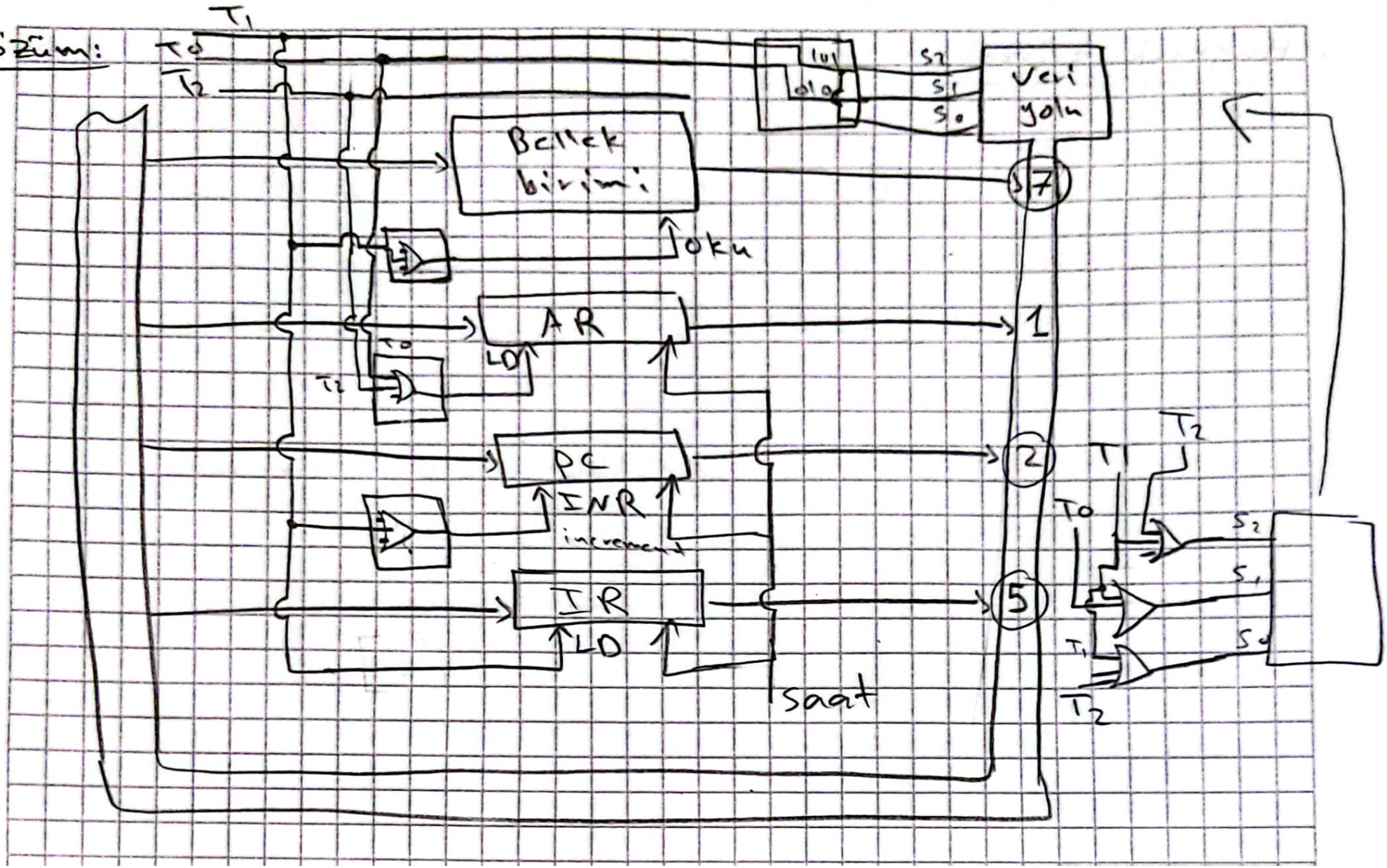
T_2: $D_0, \ldots D_7 \leftarrow$ kodu çöz $IR(12-14)$,

$AR \leftarrow IR(0-11)$,

$I \leftarrow IR(15)$

yukarıda verilen al getir evrelerini icrasını sağlayan devrenin blok şemasını aşağıda verilmiştir, Buna göre devrenin veri yolu giriş değerlerini çözüleyen dijital devreyi çiziniz.

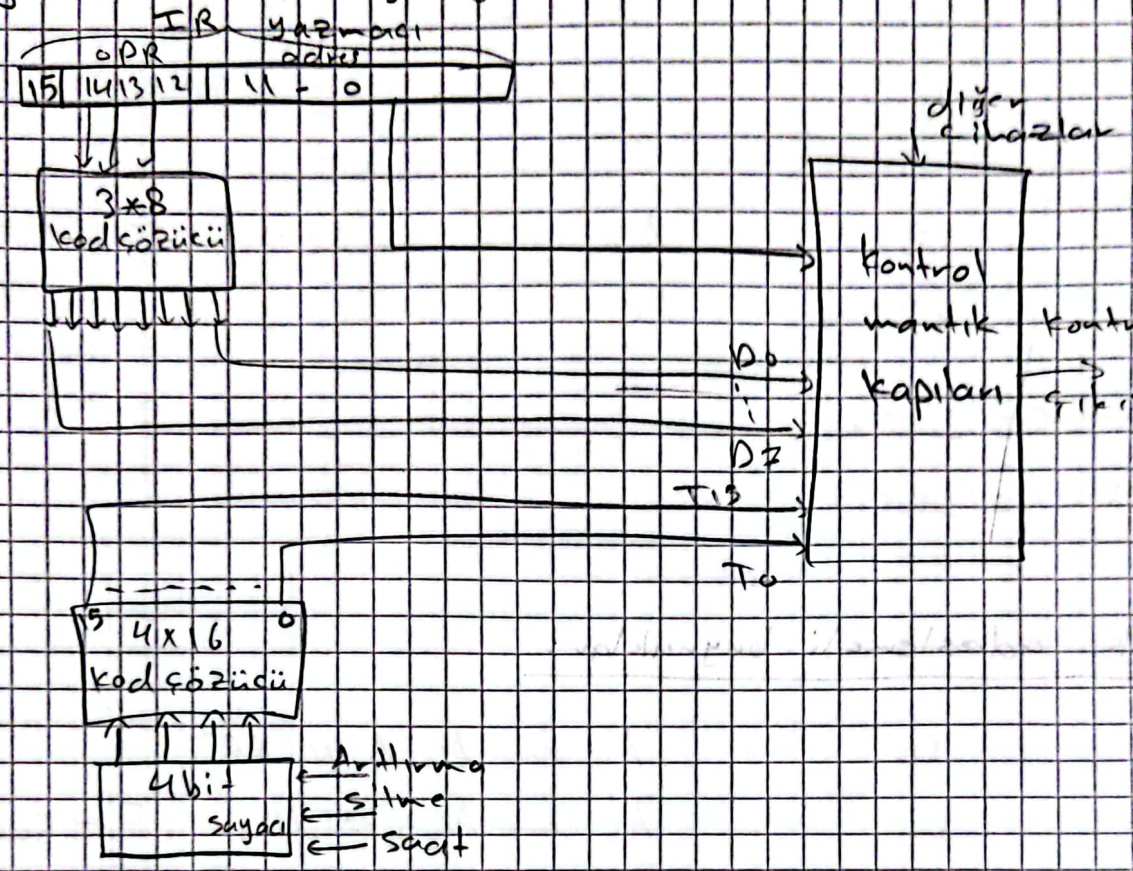
Çözüm:

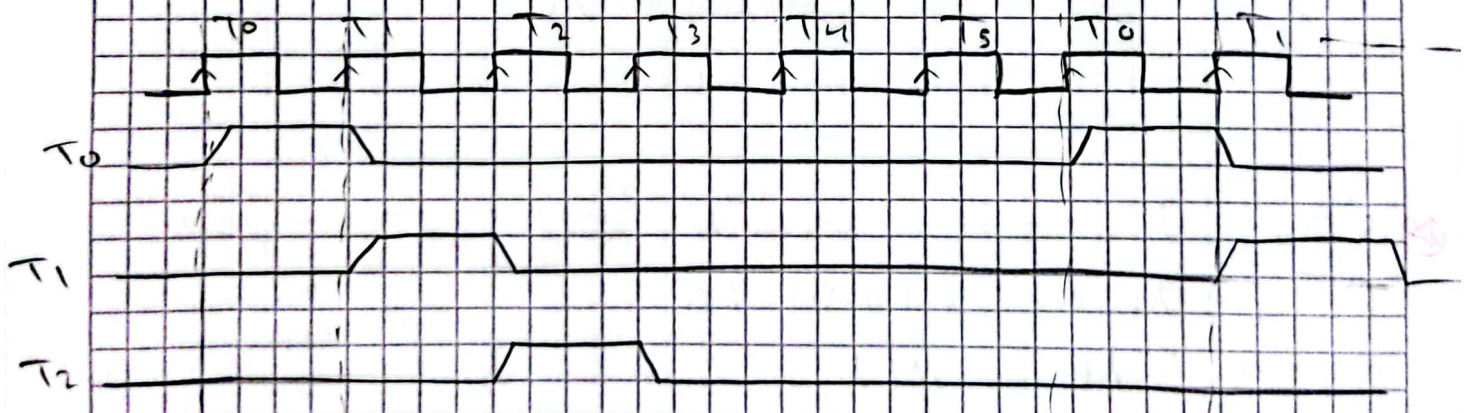

T1
T0
T2
Bellek birimi
oku
AR
LD
PC
INR
increment
TR
LD
saat
Veri yolu
S2
S1
S0
7
1
2
5
T1
T2
T0
S2
S1
S0

Hafta 14

- Bazi buyruk tipleri:

Buyruklar temel bilgi sayarın tarafından çözülür.





- Yazmaç adreslemeli buyruk tipleri:

CLE	rB_{11}	$AC \leftarrow 0$
CLA	r	$SC \leftarrow 0$
INC	rB_5	$AC \leftarrow AC + 1$
CIR	rB_7	$AC \leftarrow shr\ AC$ $\quad AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	rB_6	$AC \leftarrow shl\ AC$ $\quad AC(1) \leftarrow E, E \leftarrow AC(15)$

$r = D_7 I' T_3$

$B(i) = IR(i)$

Bellek adreslemeli buyruklar:

AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR]$, $E \leftarrow C_{çıkış}$
LAD	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
⋮	⋮	⋮

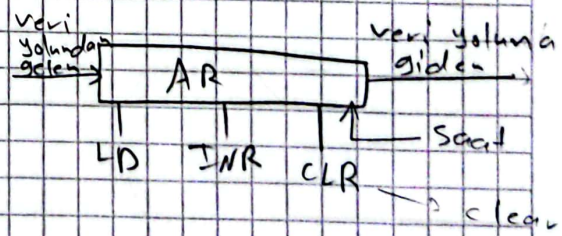
$R' T_0$: $AR \leftarrow PC$ //Load LD

$R' T_2$: $AR \leftarrow IR(0-11)$ //LD

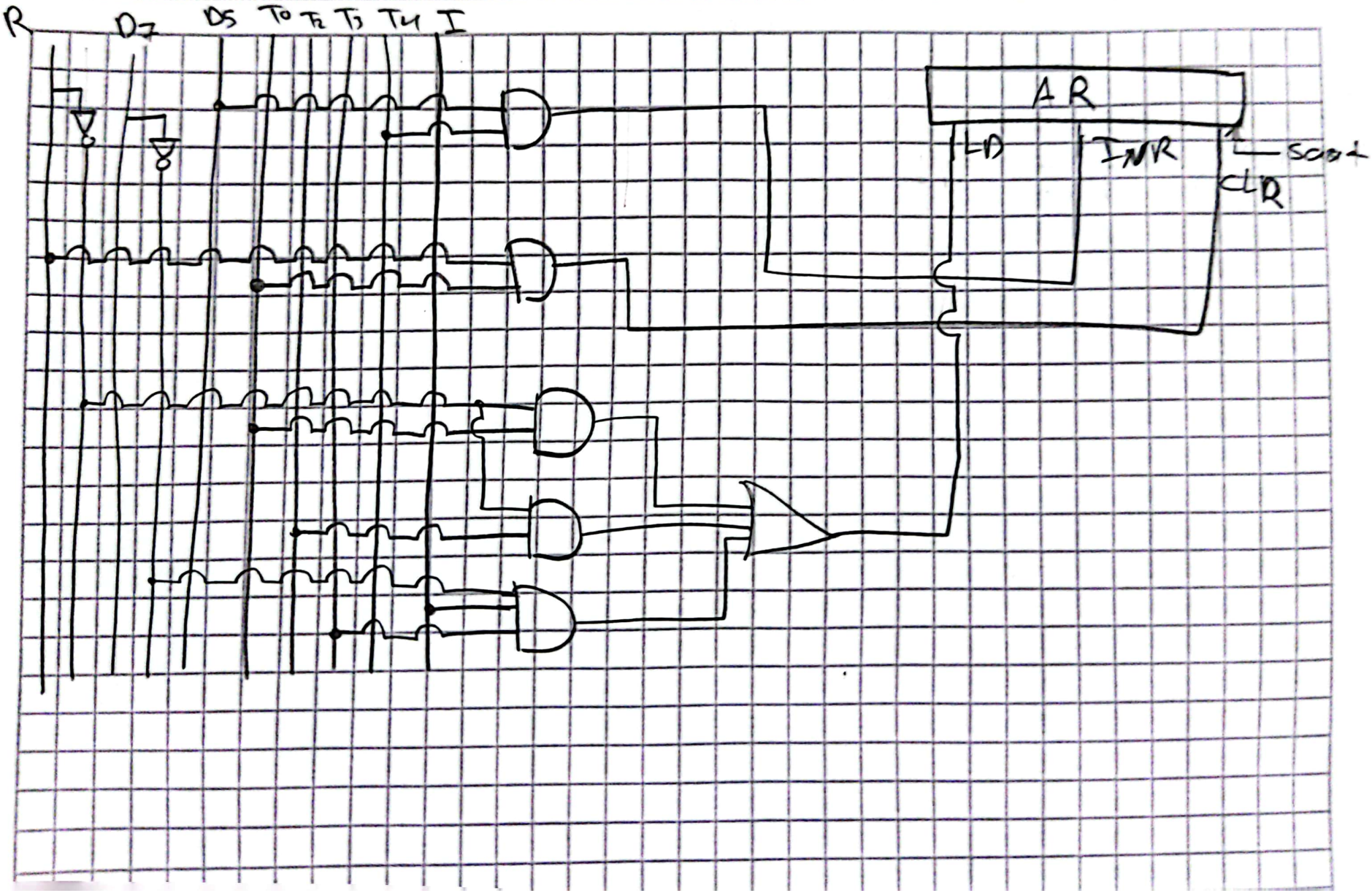
$D'_7 I T_3$: $AR \leftarrow M[AR]$ //LD

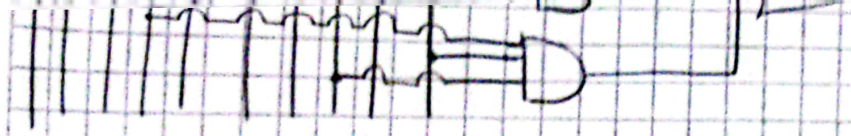
$R T_0$: $AR \leftarrow 0$ //CLR

$D_5 T_4$: $AR \leftarrow AR + 1$ //INC



verilen bir dizi işlemin gerçekleştirmesi aşamasında AR ile ilgili denetim kapıları çiziniz.


R
D7
D5
T0
T2
T3
T4
I
AR
LD
INR
CLR
scout



Bellek adreslemeli buyruklar:

kod = 000 dan 110 a kadar

15	14 … 12	11 … 0
I	kod	Adres

	I=0	I=1
AND	0XXX	8XXX
ADD	1XXX	9XXX
LDA	2XXX	AXXX
ISZ	6XXX	EXXX

Yazmaç adreslemeli buyruk

15	12 11	0
0	111	yazmaç işlemi

(kod = 111, I = 0)

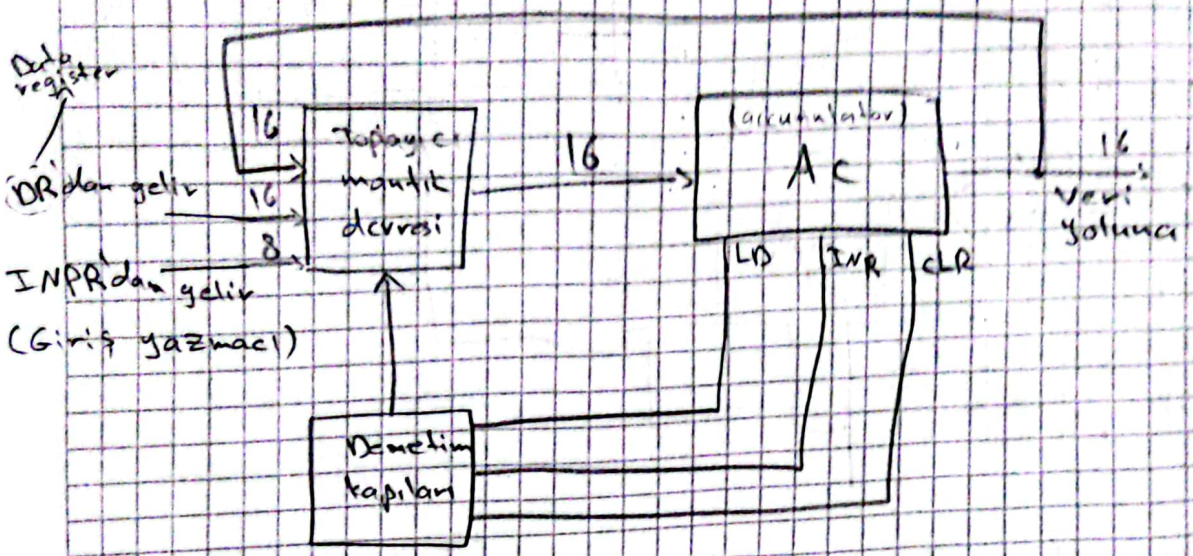
CLA	7000	AC'yi sil
CLE	7400	E'yi sil
CMA	7200	AC'nin tümleyeni sil
INC	7020	AC' değerini 1 artır.

15	12 11	0
1	111	G/Ç işlemi

(kod = 111, I = 1)

INP	F800	Giriş kanalı Giriş/çıkış rini AC'ye aktar adreslemeli
OUT	F400	AC'de çıkış karakterini al.

- <u>İşlemci yazmaç mantık tasarımı:</u>



AC ile ilişkili Devreler.

D_0T_5 : $AC \leftarrow AC \wedge DR$ //AND , LD

D_1T_4 : $AC \leftarrow AC + DR$ //ADD , LD

D_2T_5 : $AC \leftarrow DR$ // DR , LD

pB_{11} : $AC(0\text{-}7) \leftarrow INPR$ //INPR , LD

rB_9 : $AC \leftarrow \overline{AC}$ // COM (Acvsi) , LD

rB_7 : $AC \leftarrow shrAC$, $AC(15) \leftarrow E$ //ShR , LD

rB_6 : $AC \leftarrow shLAC$, $AC(0) \leftarrow E$ //ShL , LD

rB_{11} : $AC \leftarrow 0$ // CLR

rB_5 : $AC \leftarrow AC + 1$ //INC

CLA 7800 Ac'yi sil

$rB_{11} = \underset{\text{op code}}{\underline{7800\,H}}$

$B(i) = IR(i)$

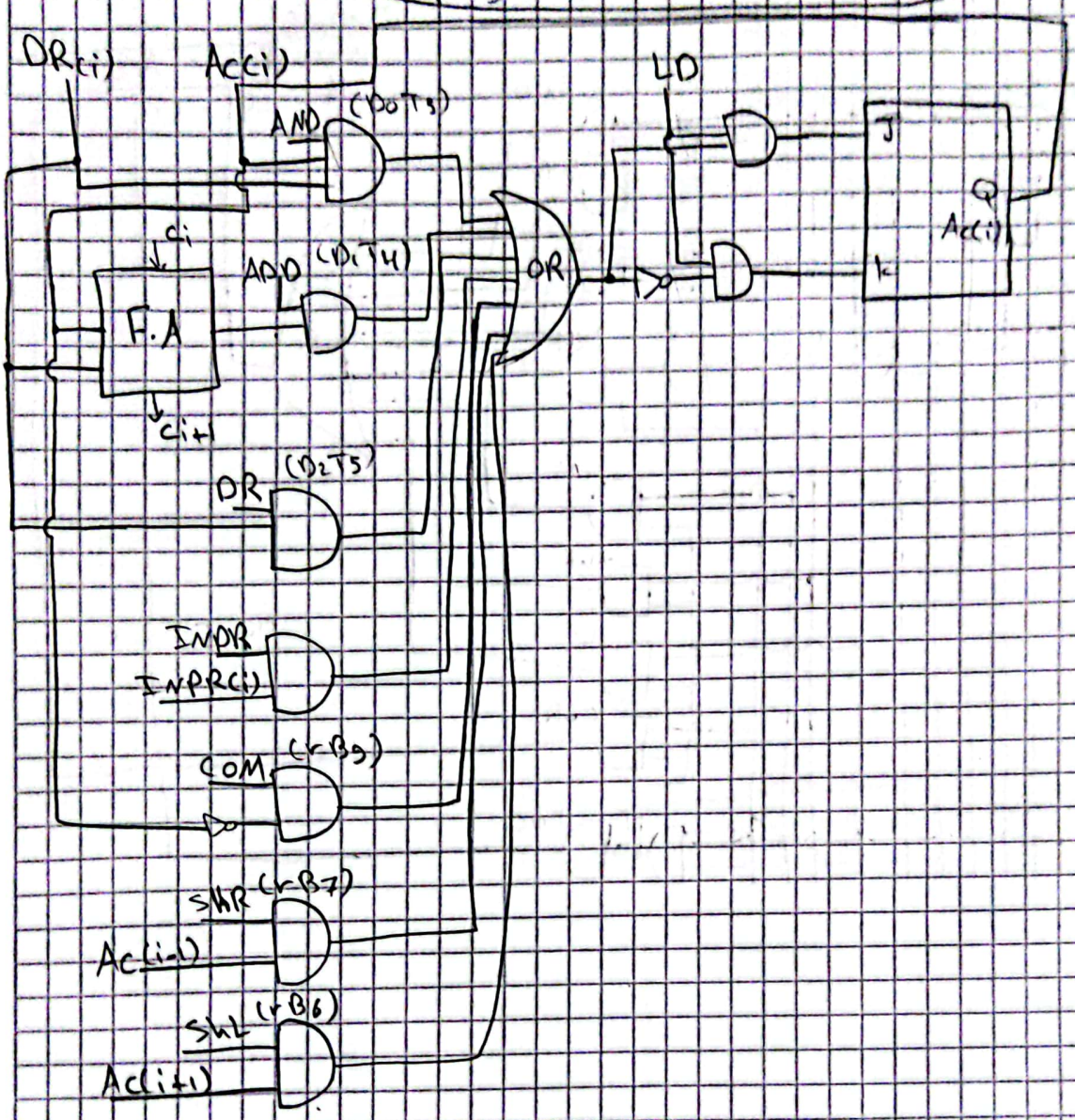
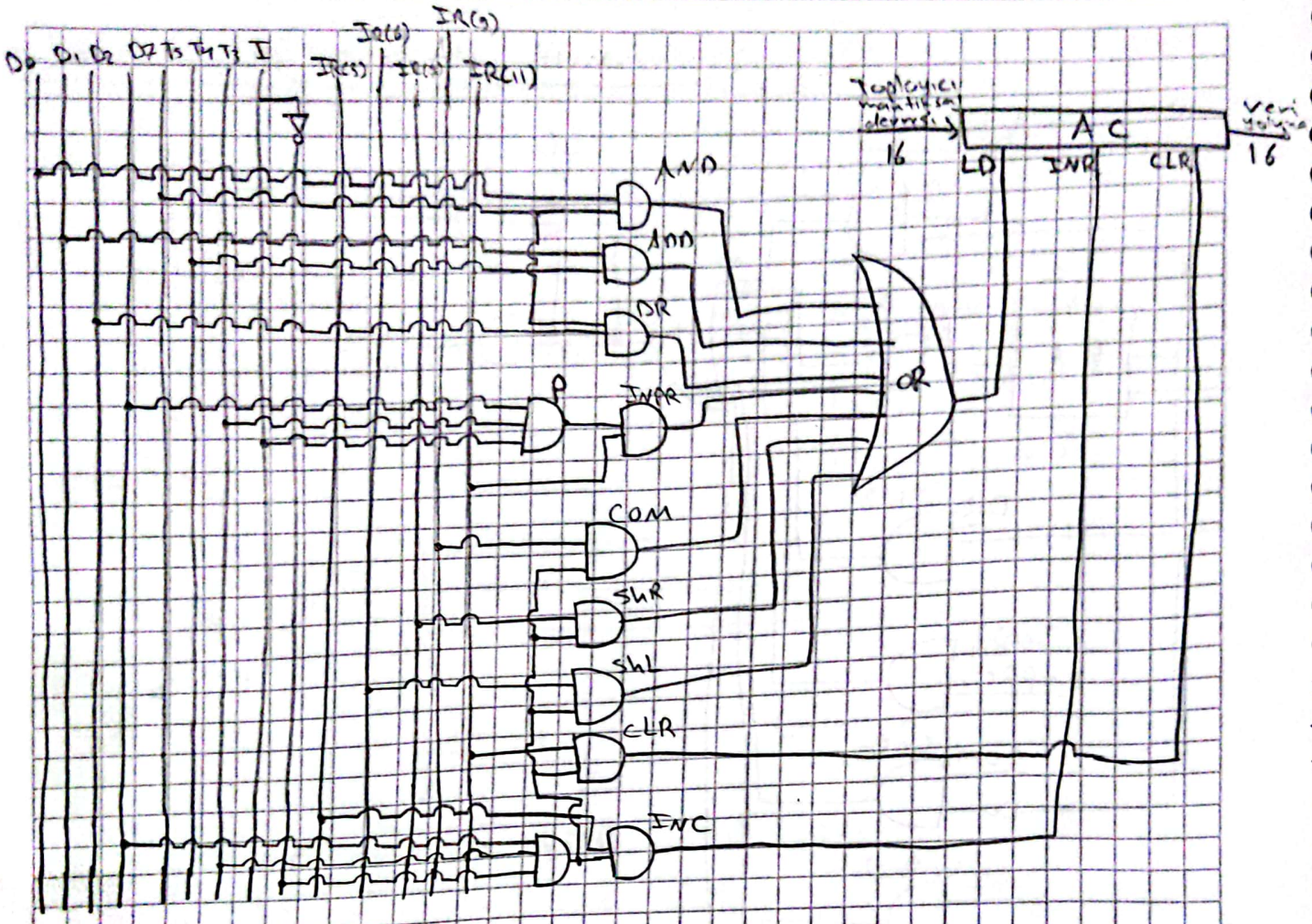
$r = D_7 I' T_3$

$p = D_7 I T_3$

$AC \leftarrow \overline{AC} + 1$ //LD (INC değil)

k girişinde D0 var çünkü JK flip-flop k girişi J'nin tersi geliyor, ondan normale dönüştürüyoruz.

Toplayıcı mantık devreleri



(5 V F.A DR+1 ISIN)