

Mikroişlemci Buyrukları

(29)

Bilgisayarların mikroişlemcisinde kullanılan komutlar, diğer bir deyişle buyruklar, 3 katagoride sınıflandırılırlar.

* Veri Aktarım Buyrukları

* Veri İşleme Buyrukları

* Program Denetim Buyrukları

Veri Aktarım Buyrukları

Veri aktarım buyrukları verileri bir yerden başka bir yere icerigini değiştirmeden gönderirler. En çok kullanılır; bellek ile işlemci yazarları arasında, işlemci yazarları ile Giriş/Gıçık birimleri arasında ve işlemci yazarlarının kendisi aralarındaki aktarımlardır.

Ld - Load (Yükle)

In - Input (Giriş)

St - Store (Aktar)

Out - Output (Gıçık)

Mov - Move (Taşı)

Push - Yığınaya Koyma

Exc - Exchange (Yer Değiştir)

Pop - Yığından Al

Veri İşleme Buyrukları

Veri işleme buyrukları veriler üzerinde işlem yaparak verilerin içeriğini değiştirirler. Üç gruba ayrılırlar.

a) Aritmetik Buyruklar (Add, Sub, Mul, Div, Inc, Dec, Neg)

b) Mantıksal Buyruklar (And, Or, Xor, Com, Clr, IntOn, IntOff)

c) Kaydırma Buyrukları (Shr, Shl, Asr, Asl, Ror, Rol)

Program Denetim Buyrukları

Buyruklar daima ardışık bellek adreslerinde bulunurlar. Mikroişlemci içinde işlenmeli işin bu buyruklar ardışık adresten alınıp getirilirler ve sonra icra edilirler. Bir buyruk adresinden alınıp getirilince PC artırlır. Böylece bir sonraki buyruğun adresi PC'de bulunur. Program denetim buyrukları düzenli olarak bu adres sırasını değiştirir.

Br - Branch (Dallan)

Cmp - Compare (Kıyasla)

Jmp - Jump (Atla)

Genelde giderken işlemiyle kıyaslamaya yarar

Call - Altprogramı Çağır

Tst - Test (Test Et)

Return - Altprogramda Dön

Genelde ve işlemi ile test eder.

Subroutine (Altrutin - Altprogram)

(26)

Belli bir hesapsal işi yapmaya yarayan program parçasıdır.

Altprograma Giriş

$M[SP] \leftarrow PC$

$SP \leftarrow SP + 1$

$PC \leftarrow$ Altprogram Adresi

Call Buyruğu

Altprogramdan Dönüş

$SP \leftarrow SP - 1$

$PC \leftarrow M[SP]$

Return Buyruğu

PC : Program Counter

SP : Stack Pointer

Yukarıda verilen üç mikro işlem Call komutu ile altprograma gidiş, diğer iki mikro işlem Return komutu ile altprogramdan dönüş işlemlerini singeler.

Interrupt (Kesinti)

İsten veya dıştan gelen bir istek sonucu, normal program akışının diğer bir servis rutinine aktarılmasıdır. Program, servis rutin bittikten sonra normal rutine geri döner.

Kesinti Rutinine Giriş

* Kosturulan Komutu bitir.

* PC ve PSW'yi yığınca aktar.

* Kesinti Rutinine git.

Kesinti Rutininden Dönüş

* RTI komutunu kostur

* PC ve PSW'yi yığından al.

* Anaprograma kaldığı yerden devam et.

Kesinti ve Altprogram arasındaki farklar

* Kesinti isten veya dıştan bir işaret ile başlar.

* Altprogram çağrılmak komutuya başlar.

* Kesintinin rutin adresleri daha önce den belirlenmiştir.

* Altprogram adresini programca belirler.

* Altprogram sadece PC bilgisini saklarken, kesinti hem PC bilgisini hem de PSW bilgisini saklar.

* PC dönüştürme programın kaldığı yeri gösterirken PSW işlemciyle ilgili en son durumlar bilgisini saklar.

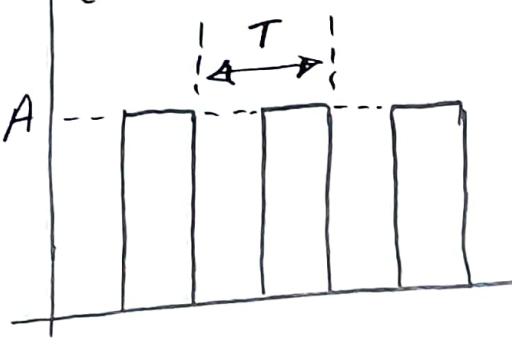
PSW : Processor Status Word
İşlemci Durum Sözcüğü

RTI : Return
From
Interrupt

Performans Ölçümleri

(27)

Amplitude
(Genlik)



$CR = \text{Clock Rate} = \text{Frekans} = f$
(Saat Hz)

$CT = \text{Clock Time} = \text{Peryod} = T = \frac{1}{f}$
(Gevrim Süresi)

$IC = \text{Instruction Count}$
(Komut Sayısı)

$CPI = \text{Cycles Per Instruction}$ (Komut Başına Düşen Gevrim)
= $\frac{\text{Total CPU Clock Cycles}}{\text{Instruction Count}} = \frac{\sum_{i=1}^n CPI_i \times I_i}{IC}$, $IC = \sum_{i=1}^n I_i$

$CC = \text{Cycle Count} = IC \times CPI$
(Gevrim Sayısı)

$ET = \text{Execution Time} = CC \times CT = IC \times CPI \times CT = \frac{IC \times CPI}{f}$

$MIPS = \text{Million Instructions Per Second}$
(Saniyedeki milyon komut sayısı)

= $\frac{\text{Instruction Count}}{\text{Execution Time} \times 10^6} = \frac{IC}{ET \times 10^6} = \frac{f}{CPI \times 10^6}$

Frekansi 200 MHz olan bir işlemci aşağıdaki komut karmaşalarını işleyen bir programı çalıştırıyor. Komut başına düşen ortalama gevrimi ve saniye başına düşen ortalama milyon komut sayısını hesapla.

Komut Kategorisi	Olmak Olasılığı	Komut Başına Düşen Gevrim Sayısı
ALU	38	1
Load/Store	15	3
Branch	42	4
Digerleri	5	5

$$IC = \sum_{i=1}^n I_i = 38 + 15 + 42 + 5 = 100$$

$$CPI = \frac{\sum_{i=1}^n CPI_i \times I_i}{IC} = \frac{38 + 15 \times 3 + 42 \times 4 + 5 \times 5}{100} = 1.00$$

$$= \frac{38 + 45 + 168 + 25}{100} = \frac{276}{100} = 2.76$$

$$MIPS = \frac{f}{CPI \times 10^6} = \frac{200 \times 10^6}{2.76 \times 10^6}$$

$$= 70.24$$

Mikroişlemciler; CISC ve RISC olmak üzere iki farklı mimari kullanırlar.

CISC - Complex Instruction Set Computer
(Karmaşık Komut Kümesi Bilgisayar)

RISC - Reduced Instruction Set Computer
(Azaltılmış Komut Kümesi Bilgisayar)

CISC Mimarının Temel Özellikleri

- * Çok sayıda komut içerir. (120 ile 350 arası)
- * Az sıklıkla kullanılan özel amaçlı komutlar içerirler.
- * Çok çeşitli adresleme modları (5 ile 20 arası)
- * Değişken uzunlukta komut formatları
- * Ana bellek referanslı komut formatları
- * Kademeli komut işleme teknigi kullanıklarından bir komutun işlenmesi bitmeden yeni bir komutu işleyemezler.
- * Mimarileri daha karmaşık olduğundan tasarımlarca daha zordur ve tasarımlarında daha fazla transistör kullanılır. Bu da gakisirken daha çok ısınmalarına neden olur.
- * Aynı işi daha az komut sırtılı kullanarak yazmak mümkündür. Fakat komutların içeriği daha yavaştır.

RISC Mimarının Temel Özellikleri

- * Daha az sayıda komut kümesi yerir. (100'den az)
- * Daha az sayıda adresleme modunu sahiptirler.
- * İşlemler yazılımlarda yapıldığından daha hızlıdır.
- * Ana bellek erişimi Load ve Store komutlarıyla gerçekleştirir.
- * Sabit uzunluklu kolay işlenebilen komut formatları kullanırlar.
- * Komutlar daha basit ve komut sayısı daha az olduğundan işlemci tasarımları daha kolaydır.
- * Kanal komut işleme (pipeline) teknigi kullanıkları işin aynı anda çok sayıda komut işleyebilirler. Bir komutun başında iken diğer bir komutun sonunda olabildikleri işin her işlem adımında bir komuta ait işi bitirler. Bu durumda her gevrimde bir komut icra edilmiş olur.
- * Daha az sayıda transistör kullanıldığı için daha az ısınır. Soğutma için daha az enerji kullanılır.

$F \leftarrow A * B - C / (D + E)$ fonksiyonu ile verilen
 işlemi yapan programı aşağıdaki şıklara göre yazınız. (29)
 a) Sıfır adres buyruklu komut kümesi olan bir dille
 b) Bir adres buyruklu komut kümesi olan bir dille
 c) İki adres buyruklu komut kümesi olan ve CISC
 mimarı kullanan bir dille
 d) İki adres buyruklu komut kümesi olan ve RISC
 mimarı kullanan bir dille

a) Push A
 Push B
 Mul
 Push C
 Push D
 Push E
 Add
 Div
 Sub
 Pop F

b) Load D ($Acc \leftarrow M[D]$)
 Add E ($Acc \leftarrow Acc + M[E]$)
 Store F ($M[F] \leftarrow Acc$)
 Load C ($Acc \leftarrow M[C]$)
 Div F ($Acc \leftarrow Acc / M[F]$)
 Store F ($M[F] \leftarrow Acc$)
 Load A ($Acc \leftarrow M[A]$)
 Mul B ($Acc \leftarrow Acc * M[B]$)
 Sub F ($Acc \leftarrow Acc - M[F]$)
 Store F ($M[F] \leftarrow Acc$)

c) CISC Mimari

Mov R1, A ($R1 \leftarrow M[A]$)
 Mul R1, B ($R1 \leftarrow R1 * M[B]$)
 Mov R2, C ($R2 \leftarrow M[C]$)
 Mov R3, D ($R3 \leftarrow M[D]$)
 Mov R3, E ($R3 \leftarrow R3 + M[E]$)
 Add R3, E ($R3 \leftarrow R3 / R2$)
 Div R2, R3 ($R2 \leftarrow R2 - R3$)
 Sub R1, R2 ($R1 \leftarrow R1 - R2$)
 Mov F, R1 ($M[F] \leftarrow R1$)

d) RISC Mimari

Load R1, A ($R1 \leftarrow M[A]$)
 Load R2, B ($R2 \leftarrow M[B]$)
 Mul R1, R2 ($R1 \leftarrow R1 * R2$)
 Load R2, C ($R2 \leftarrow M[C]$)
 Load R3, D ($R3 \leftarrow M[D]$)
 Load R4, E ($R4 \leftarrow M[E]$)
 Add R3, R4 ($R3 \leftarrow R3 + R4$)
 Div R2, R3 ($R2 \leftarrow R2 / R3$)
 Sub R1, R2 ($R1 \leftarrow R1 - R2$)
 Store F, R1 ($M[F] \leftarrow R1$)

Acc - Accumulator
 Biriker Vega İşlemci Yazılımı
 Tek yazısı var

$$F \leftarrow (A+B)/C - D * E$$

Fonksiyonu ile verilen ifadesi (30)

yapan programı aşağıdaki şıklara göre yazınız.

- a) Sıfır adres buyruklu komut kümesi olan bir dille
- b) Bir adres buyruklu komut kümesi olan bir dille
- c) İki adres buyruklu komut kümesi olan CISC mimari kullanan bir dille
- d) İki adres buyruklu komut kümesi olan RISC mimari kullanan bir dille

a) Push A
Push B
Add
Push C
Div
Push D
Push E
Mul
Sub
Pop F

b) Load D ($Acc \leftarrow M[D]$)
Mul E ($Acc \leftarrow Acc * M[E]$)
Store F ($M[F] \leftarrow Acc$)
Load A ($Acc \leftarrow M[A]$)
Add B ($Acc \leftarrow Acc + M[B]$)
Div C ($Acc \leftarrow Acc / M[C]$)
Sub F ($Acc \leftarrow Acc - M[F]$)
Store F ($M[F] \leftarrow Acc$)

c) CISC Mimari

MOV R1,A ($R1 \leftarrow M[A]$)
Add R1,B ($R1 \leftarrow R1 + M[B]$)
Div R1,C ($R1 \leftarrow R1 / M[C]$)
MOV R2,D ($R2 \leftarrow M[D]$)
Mul R2,E ($R2 \leftarrow R2 * M[E]$)
Sub R1,R2 ($R1 \leftarrow R1 - R2$)
Mov F,R1 ($M[F] \leftarrow R1$)

d) RISC Mimari
Load R1,A ($R1 \leftarrow M[A]$)
Load R2,B ($R2 \leftarrow M[B]$)
Add R1,R2 ($R1 \leftarrow R1 + R2$)
Load R2,C ($R2 \leftarrow M[C]$)
Div R1,R2 ($R1 \leftarrow R1 / R2$)
Load R2,D ($R2 \leftarrow M[D]$)
Load R3,E ($R3 \leftarrow M[E]$)
Mul R2,R3 ($R2 \leftarrow R2 * R3$)
Sub R1,R2 ($R1 \leftarrow R1 - R2$)
Store F,R1 ($M[F] \leftarrow R1$)

$$Y \leftarrow (A + B * C) / (D - E * F + G * H)$$

Sıfır Adres
Buyruklu
Komut kimesi
olan bir dille

Push A

Push B

Push C

Mul

Add

Push D

Push E

Push F

Mul

Sub

Push G

Push H

Mul

Add

Div

Pop Y

Bir adres buyruklu komut kimesi
olan bir dille

Load E ($Acc \leftarrow M[E]$)

Mul F ($Acc \leftarrow Acc * M[F]$)

Store Y ($M[Y] \leftarrow Acc$)

Load D ($Acc \leftarrow M[D]$)

Sub Y ($Acc \leftarrow Acc - M[Y]$)

Store Y ($M[Y] \leftarrow Acc$)

Load G ($Acc \leftarrow M[G]$)

Mul H ($Acc \leftarrow Acc * M[H]$)

Add Y ($Acc \leftarrow Acc + M[Y]$)

Store Y ($M[Y] \leftarrow Acc$)

Load B ($Acc \leftarrow M[B]$)

Mul C ($Acc \leftarrow Acc * M[C]$)

Add A ($Acc \leftarrow Acc + M[A]$)

Div Y ($Acc \leftarrow Acc / M[Y]$)

Store Y ($M[Y] \leftarrow Acc$)

Mul R1, B, C ($R1 \leftarrow M[B] * M[C]$)

Add R1, A, R1 ($R1 \leftarrow M[A] + R1$)

Mul R2, E, F ($R2 \leftarrow M[E] * M[F]$)

Sub R2, D, R2 ($R2 \leftarrow M[D] - R2$)

Mul R3, G, H ($R3 \leftarrow M[G] * M[H]$)

Add R2, R2, R3 ($R2 \leftarrow R2 + R3$)

Div Y, R1, R2 ($M[Y] \leftarrow R1 / R2$)

CISC mimariye göre
gözüml.

Üç adres
buyruklu
komut
kimesi
olan bir
dille

iki adres buyruklu komut komesi olan bir dille (32)

$$Y = (A + B * C) / (D - E * F + G * H)$$

CISC Mimariye göre

Mov R1, B ($R1 \leftarrow M[B]$)
Mul R1, C ($R1 \leftarrow R1 * M[C]$)
Add R1, A ($R1 \leftarrow R1 + M[A]$)
Mov R2, D ($R2 \leftarrow M[D]$)
Mov R3, E ($R3 \leftarrow M[E]$)
Mul R3, F ($R3 \leftarrow R3 * M[F]$)
Sub R2, R3 ($R2 \leftarrow R2 - R3$)
Mov R3, G ($R3 \leftarrow M[G]$)
Mul R3, H ($R3 \leftarrow R3 * M[H]$)
Add R2, R3 ($R2 \leftarrow R2 + R3$)
Div R1, R2 ($R1 \leftarrow R1 / R2$)
Mov Y, R1 ($M[Y] \leftarrow R1$)

$$Y = \frac{A - B + C * (D * E - F)}{G + H * I}$$

Infix $a+b$ (İş ek)
Prefix $+ab$ (Ön ek)
Postfix $ab+$ (Son ek) Postfix
 $AB - C D E * F - * G H I * + /$

Sıfır adres buyruklu
komut komesi olan
bir dil ile programı
yazalım.

RISC Mimariye göre

Load R1, A ($R1 \leftarrow M[A]$)
Load R2, B ($R2 \leftarrow M[B]$)
Load R3, C ($R3 \leftarrow M[C]$)
Mul R2, R3 ($R2 \leftarrow R2 * R3$)
Add R1, R2 ($R1 \leftarrow R1 + R2$)
Load R2, D ($R2 \leftarrow M[D]$)
Load R3, E ($R3 \leftarrow M[E]$)
Load R4, F ($R4 \leftarrow M[F]$)
Mul R3, R4 ($R3 \leftarrow R3 * R4$)
Sub R2, R3 ($R2 \leftarrow R2 - R3$)
Load R3, G ($R3 \leftarrow M[G]$)
Load R4, H ($R4 \leftarrow M[H]$)
Mul R3, R4 ($R3 \leftarrow R3 * R4$)
Add R2, R3 ($R2 \leftarrow R2 + R3$)
Div R1, R2 ($R1 \leftarrow R1 / R2$)
Store Y, R1 ($M[Y] \leftarrow R1$)

Push A
Push B
Sub
Push C
Push D
Push E
Mul
Push F
Sub
Mul
Push G
Push H
Push I
Mul
Add
Div
Pop Y

Adresleme Modları

33

Load Rn, #A	$R_n \leftarrow A$	Immediate (Derhal, Hemen, Acil)
Load Rn, A	$R_n \leftarrow M[A]$	Direct (Dolaysız, Doğrudan)
Load Rn, (A)	$R_n \leftarrow M[M[A]]$	Indirect (Dolaylı)
Load Rn, Rm	$R_n \leftarrow R_m$	Register Direct
Load Rn, (Rm)	$R_n \leftarrow M[R_m]$	Register Indirect
Load Rn, X(Rm)	$R_n \leftarrow M[R_m + X]$	Indexed (İndekslenmiş)
Load Rn, X(PC)	$R_n \leftarrow M[PC + X]$	Relative (İzafi, Bağlı)

Derhal Adresleme



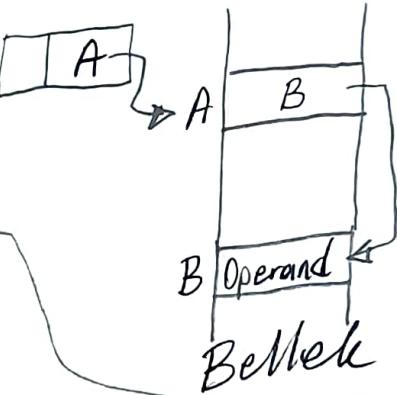
Operand (İstelenen)

Dolaysız Adresleme



Bellek

Dolaylı Adresleme



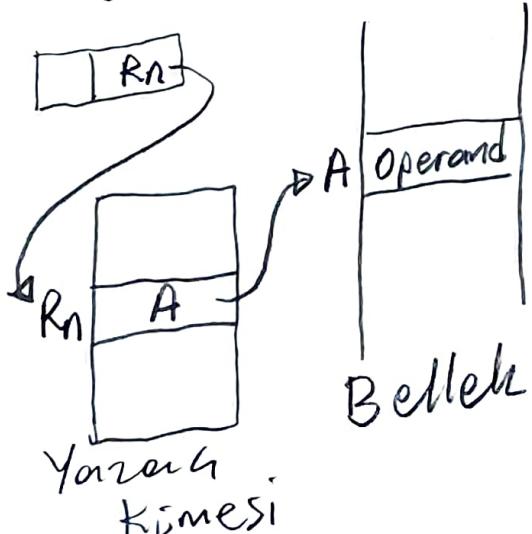
Bellek

Doğrudan Yazar Adresleme

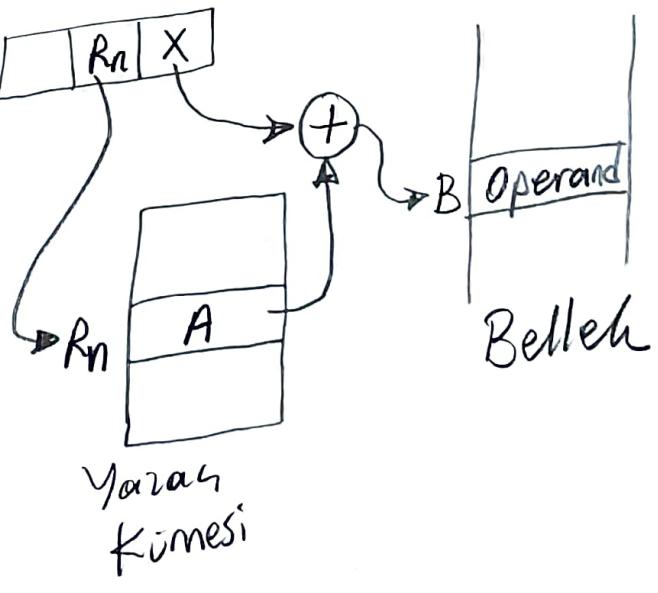


Register Set
(Yazar Kumesi)

Dolaylı Yazar Adresleme



İndekslenmiş Adresleme



Yazar
Kumesi

20	40
30	50
40	60
50	70

↓ ↓
Adresler Değerler

Bir adresli komut formath bir bilgisayar için aşağıdaki komutları inceleyelim.

$$\text{Load } \#20 \Rightarrow \text{Acc} \leftarrow 20$$

$$\text{Load } 20 \Rightarrow \text{Acc} \leftarrow M[20] = 40$$

$$\text{Load } (20) \Rightarrow \text{Acc} \leftarrow M[M[20]] = 60$$

$$\text{Load } \#30 \Rightarrow \text{Acc} \leftarrow 30$$

$$\text{Load } 30 \Rightarrow \text{Acc} \leftarrow M[30] = 50$$

$$\text{Load } (30) \Rightarrow \text{Acc} \leftarrow M[M[30]] = 70$$

Bilgisayarımız 11 bitlik komut formata sahip olsun.
Adresleme alanı 4 bit olsun. Bilgisayar genişleyebilen
Opcode teknigi kullanınsın. 5 tane iki adresli komuta,
32 tane bir adresli komuta sahip olsun
Bilgisayarın sıfır adresli komut sayısı maksimum nedir?

$$\text{Toplam Komut Sayısı} = 2^{11} = 2048$$

$$\text{Adresleme Alanı} = 4$$

$$\text{i}ki \text{ Adresli Komut için} \quad 5 \times 2^4 \times 2^4 = 1280$$

$$\text{Bir Adresli Komut için} \quad 32 \times 2^4 = 512$$

$$\text{Sıfır Adresli Komut için} \quad 2048 - 1280 - 512 = 256$$

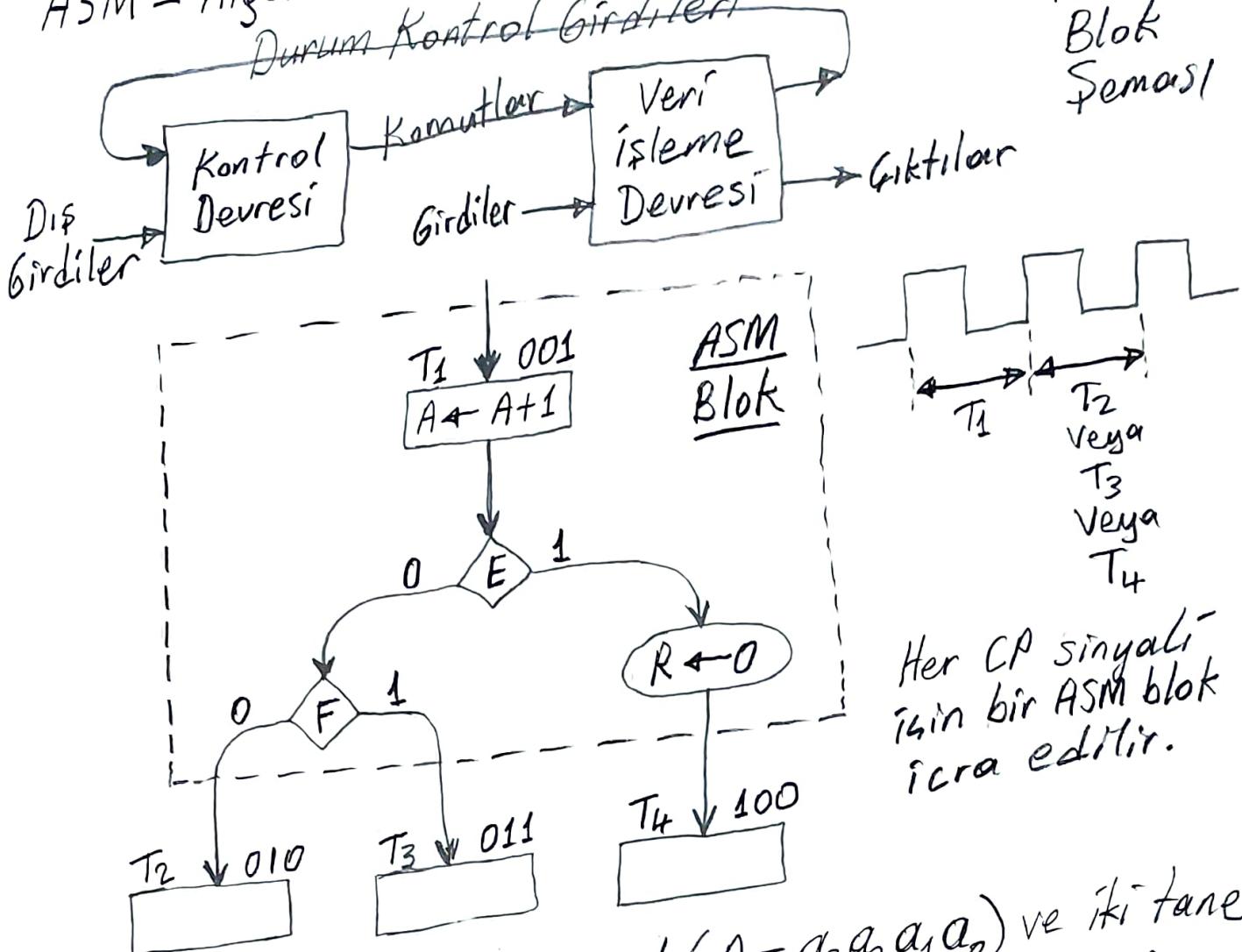
Sıfır Adresli Maksimum
komut sayısı 256'dır.

Algoritmik Durum Makineleri

ASM - Algorithmic State Machine

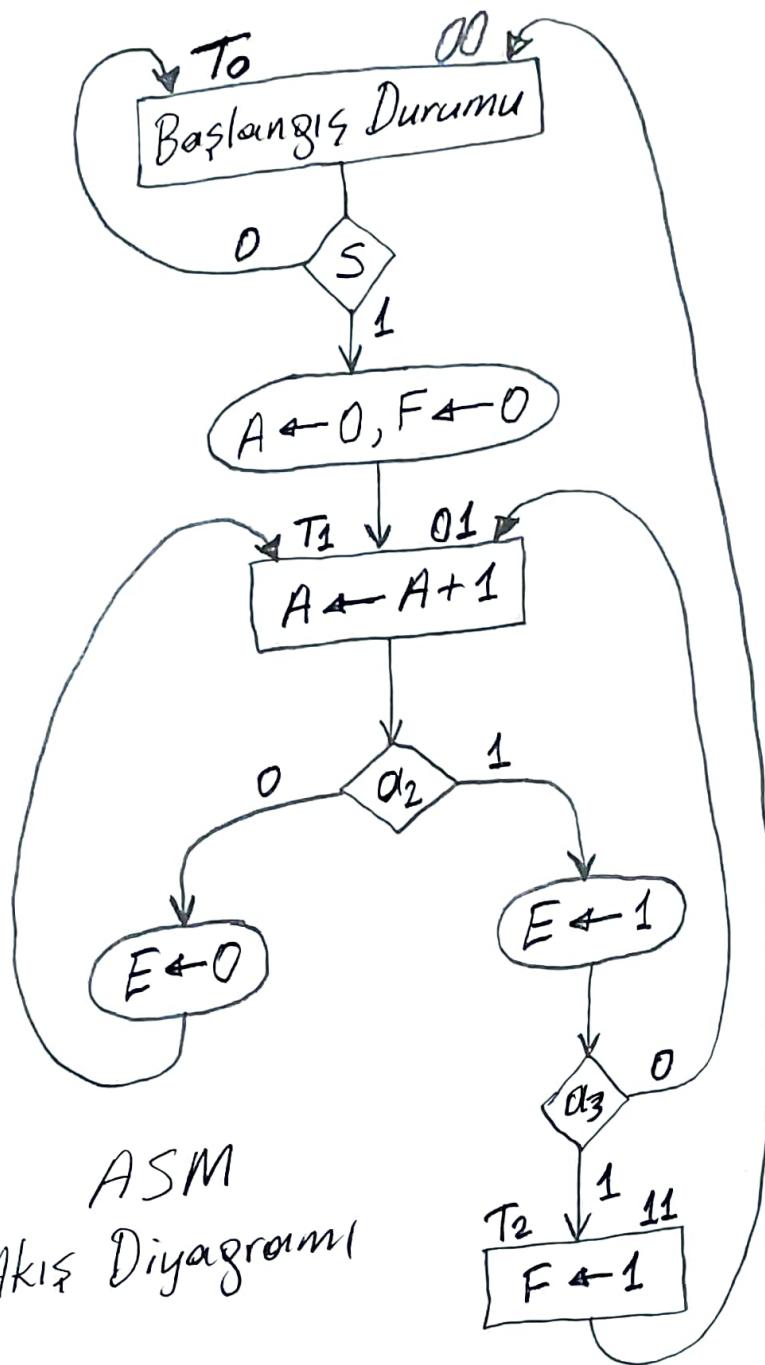
Durum Kontrol Girişleri

ASM
Blok
Sembol



Her CP sinyali
isin bir ASM blok
icra edilir.

Ür Bir tane 4 bitlik sayıcı ($A = a_3a_2a_1a_0$) ve iki tane flip-floplu (E ve F) bir sayısal sistem düşünelim. Sistem 5 başlama sinyaliyle A sayıcısını ve F flip-flop'unu sıfırlayarak çalışmaya başlıyor. Sayma işlemi durana kadar her CP sinyali isin A sayıcısı 1 artar. (0 ile 15 arası) $a_2=0$ ise $E=0$ olur ve sayma işlemine devam eder $a_2=1$ ise $E=1$ olur ve a_3 bitinin durumuna göre $a_4=0$ ise sayma işlemine devam, degilse diger CP sinyali isin $F=1$ olur ve sayma işlemi durur.



$$ST_0: A \leftarrow 0, F \leftarrow 0$$

$$T_1: A \leftarrow A + 1$$

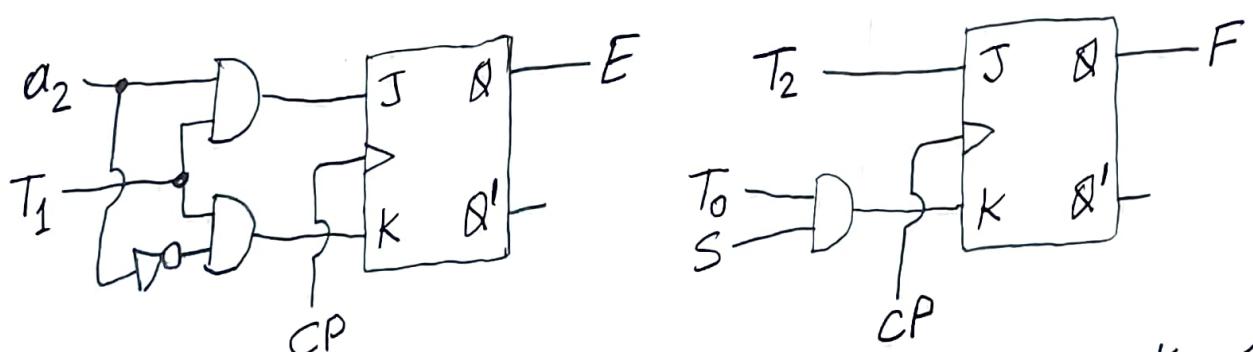
$$a'_2 T_1: E \leftarrow 0$$

$$a_2 T_1: E \leftarrow 1$$

$$T_2: F \leftarrow 1$$

JK	Q_{t+1}
00	Q_t
01	0
10	1
11	Q'_t

JK flip-flop Karakteristik Tablosu



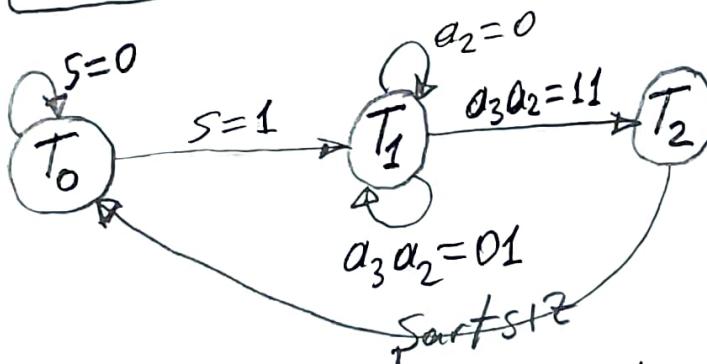
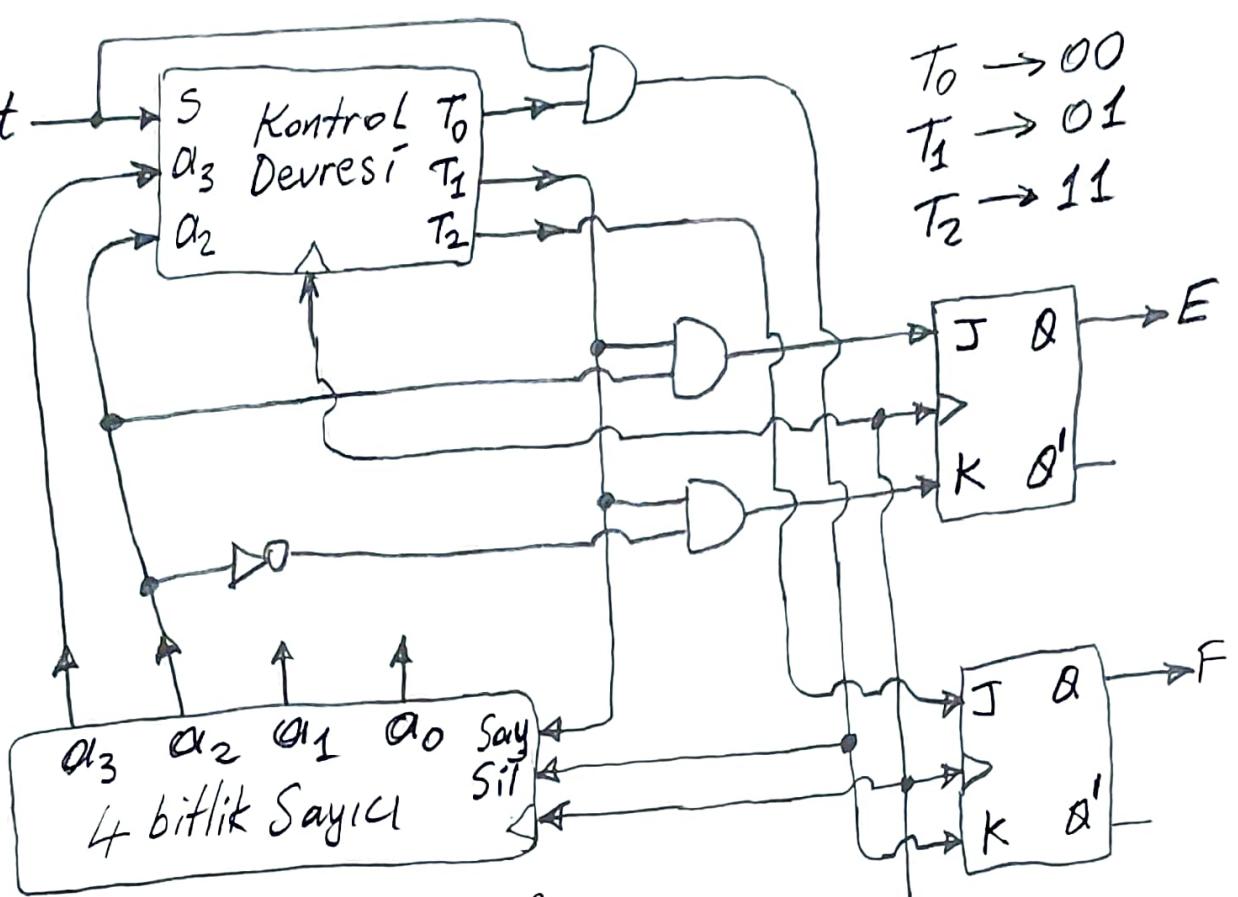
T_1 olduğunda

$a_2 = 0$ ise $J=0, K=1$

$a_2 = 1$ ise $J=1, K=0$

$ST_0 = 1$ iken $J=0, K=1$

$T_2 = 1$ iken $J=1, K=0$



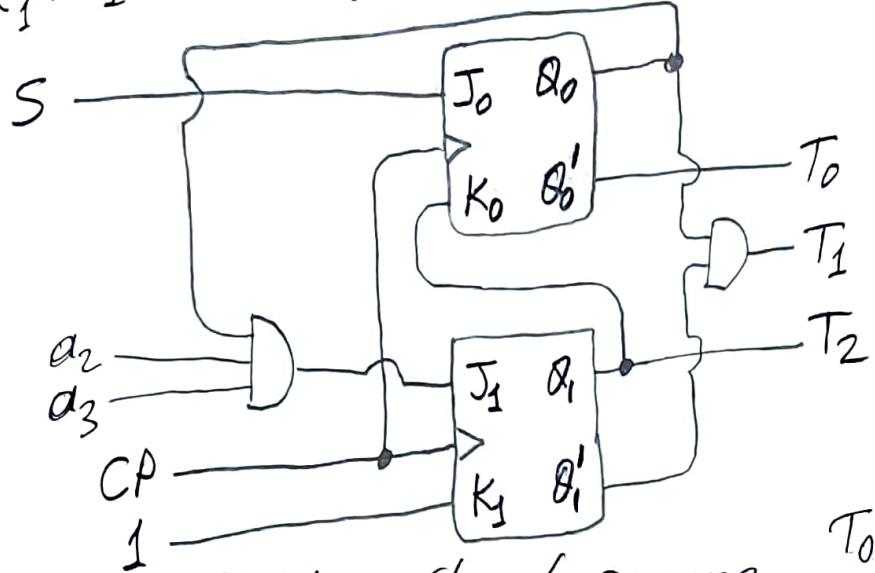
Kontrol Devresinin
Durum Geçiş
Diyagramı

PS	PS	Girişler	NS	Göküşler	JK flip-flop	D flip-flop	
	$\alpha_1\alpha_0$	$S\alpha_3\alpha_2$	$\alpha_1\alpha_0$	$T_0 T_1 T_2$	$J_1 K_1$	$J_0 K_0$	$D_1 D_0$
T_0	00	0XX	00	100	0X	0X	0 0
T_0	00	1XX	01	100	0X	1X	0 1
T_1	01	XX0	01	010	0X	X0	0 1
T_1	01	X01	01	010	0X	X0	0 1
T_1	01	X11	11	010	1X	X0	1 1
T_2	11	XXX	00	001	X1	X1	0 0

Durum Geçiş Tablosu

JK flip-flop'lar ile tasarım

$$J_1 = \bar{Q}_0 a_3 a_2 \quad J_0 = 1 \quad K_0 = Q_1 \quad T_0 = Q'_0, \quad T_1 = Q'_1 Q_0, \quad T_2 = Q_1$$

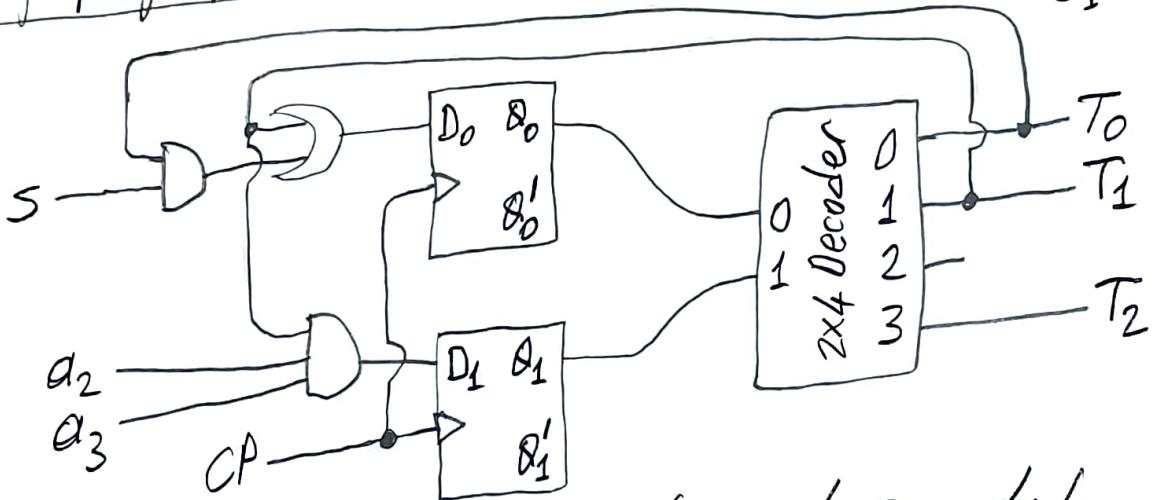


(38)

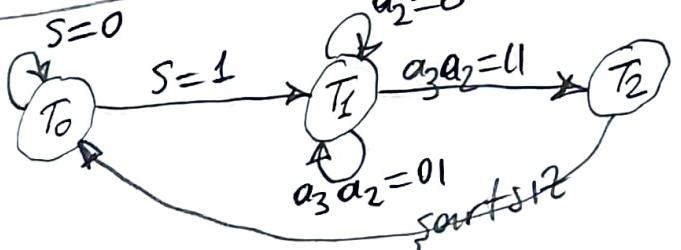
JK flip-flop'lar
ile kontrol
devresinin
tasarımı

D flip-flop'lar ile tasarım

$$T_0 = Q'_1 Q'_0 \quad T_2 = Q_1 Q_0 \\ T_1 = Q'_1 Q_0 \quad D_0 = a_3 a_2 T_1 \\ D_1 = S T_0 + T_1$$



Üç tane D flip-flop ile tasarlanısaydık

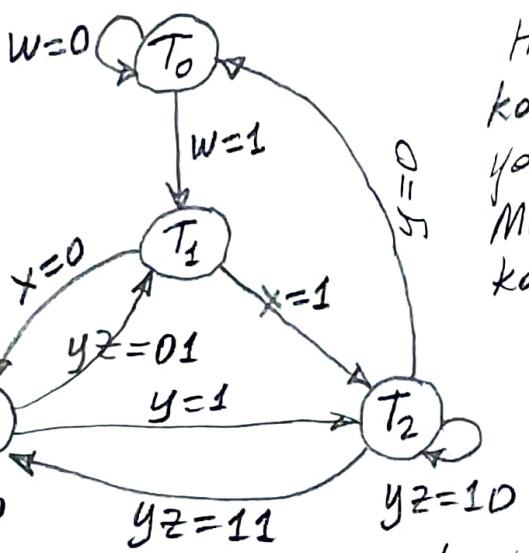


Kontrol devresinin
durum geçiş
diagramının
kullanıldık

$$D_0 = S' T_0 + T_2 \quad D_1 = S T_0 + a'_2 T_1 + a'_3 a_2 T_1 \\ = S T_0 + (a_3 a_2)' T_1$$

$$D_2 = a_3 a_2 T_1$$

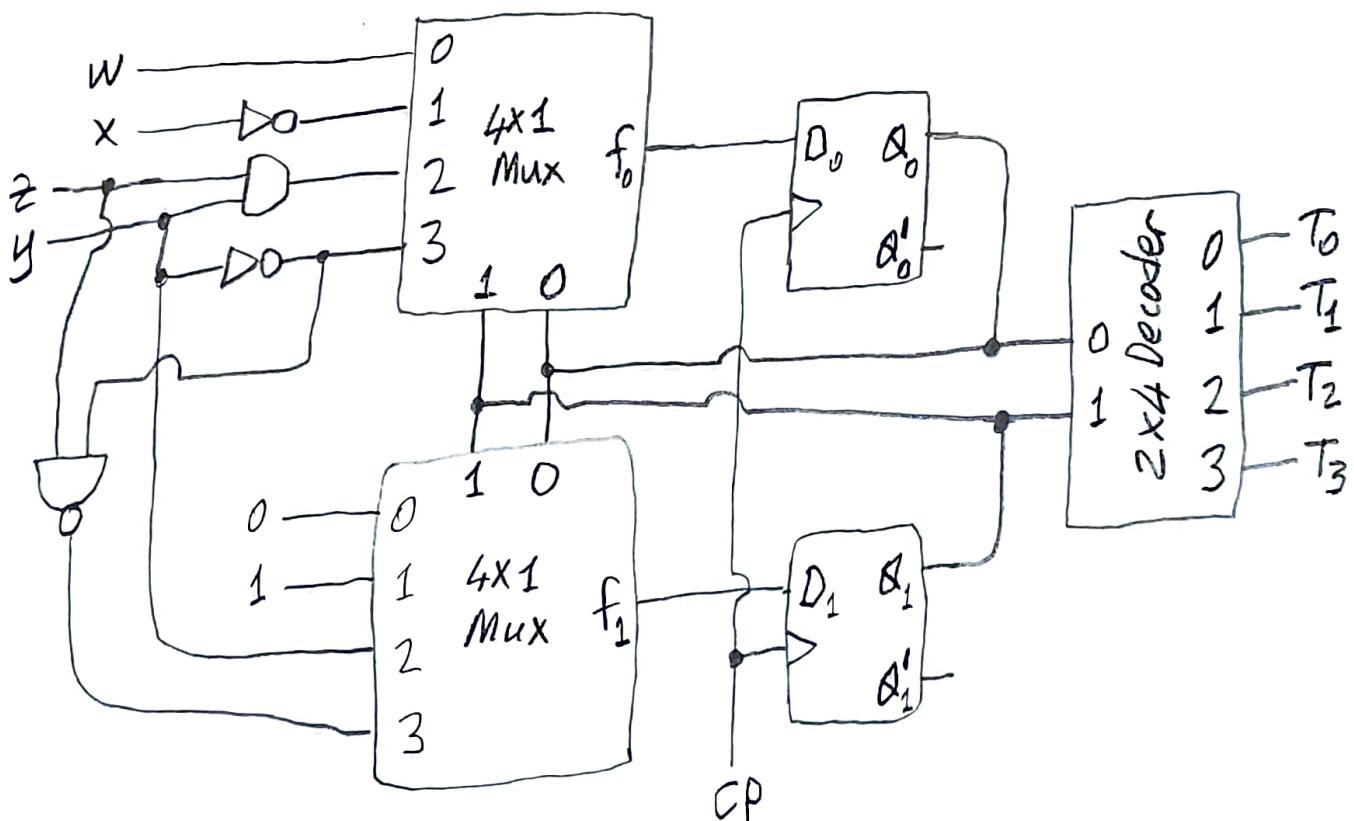
$$T_0 = Q_0, \quad T_1 = Q_1, \quad T_2 = Q_2$$



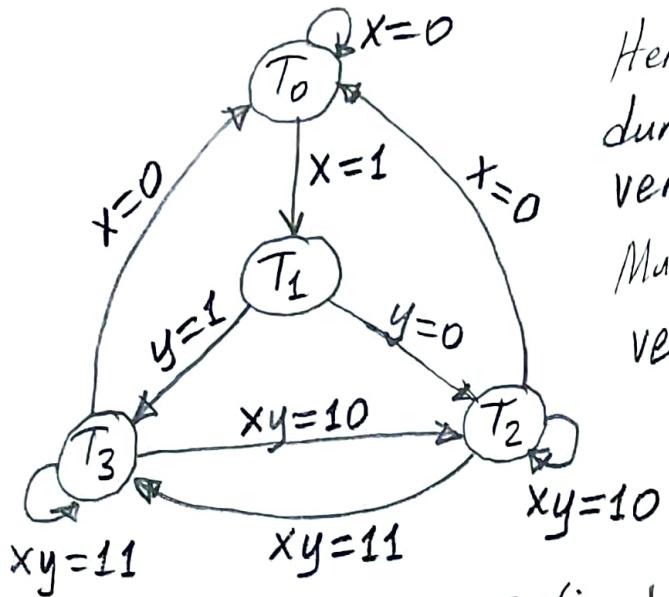
Herhangi bir ASM devresinin kontrol kısmının durum diyagramı yanında verilmiştir. Kontrol devresini Multiplexer, Decoder, D flip-flop ve kapı elementleri kullanarak tasarlayın. (39)

$$\begin{array}{ll} T_0 = 00 & T_2 = 10 \\ T_1 = 01 & T_3 = 11 \end{array}$$

$\bar{Q}_1 Q_0$	$Q_1 Q_0$	Girdi Sartları	Mux 1	Mux 0
00	00	w'	0	w
00	01	w		
01	10	x	1	x'
01	11	x'		
10	00	y'	$yz' + yz$	
10	10	yz'	$=y$	yz
10	11	yz		
11	01	$y'z$	$y + y'z'$	$y'z + y'z'$
11	10	y	$=y + z'$	$=y'$
11	11	$y'z'$	$=(y'z)'$	

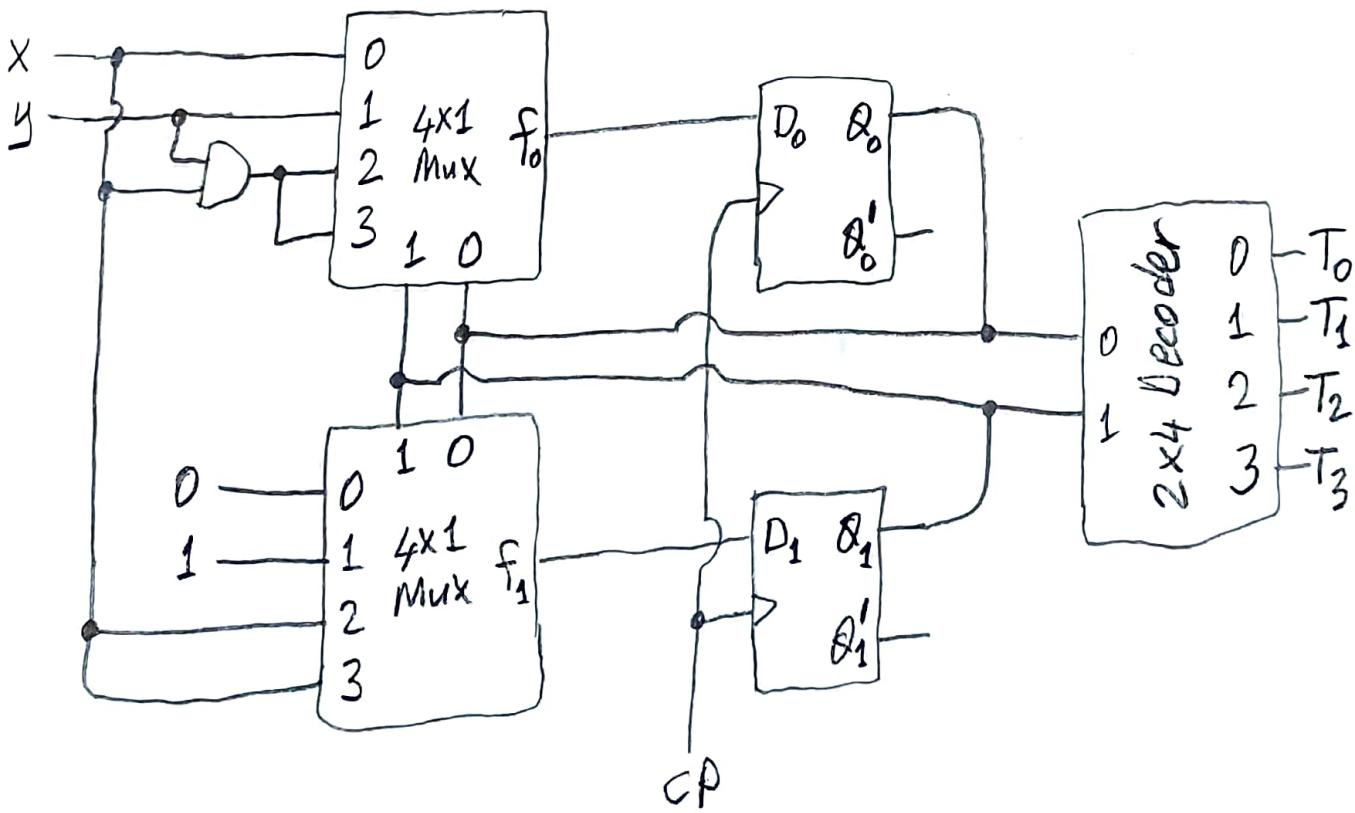


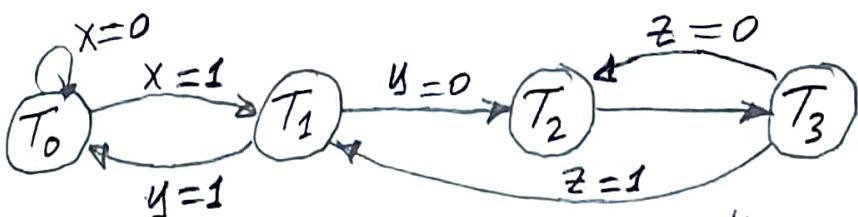
Herhangi bir ASM devresinin durum diyagramı yanda verilmiştir. Kontrol devresini Multiplexer, Decoder, D flip-flop ve kapı elementleri kullanarak tasarlanyınız.



$$\begin{array}{ll}
 T_0 = 00 & T_2 = 10 \\
 T_1 = 01 & T_3 = 11
 \end{array}$$

$Q_1 Q_0$	$Q_1 Q_0$	Girdi Sinyolları	Mux. Girişleri	Mux1	Mux0
00	00	x'	0	x	
00	01	x			
01	10	y'	1	y	
01	11	y			
10	00	x'			
10	10	xy'	x	xy	
10	11	xy			
11	00	x'			
11	10	xy'	x	xy	
11	11	xy			

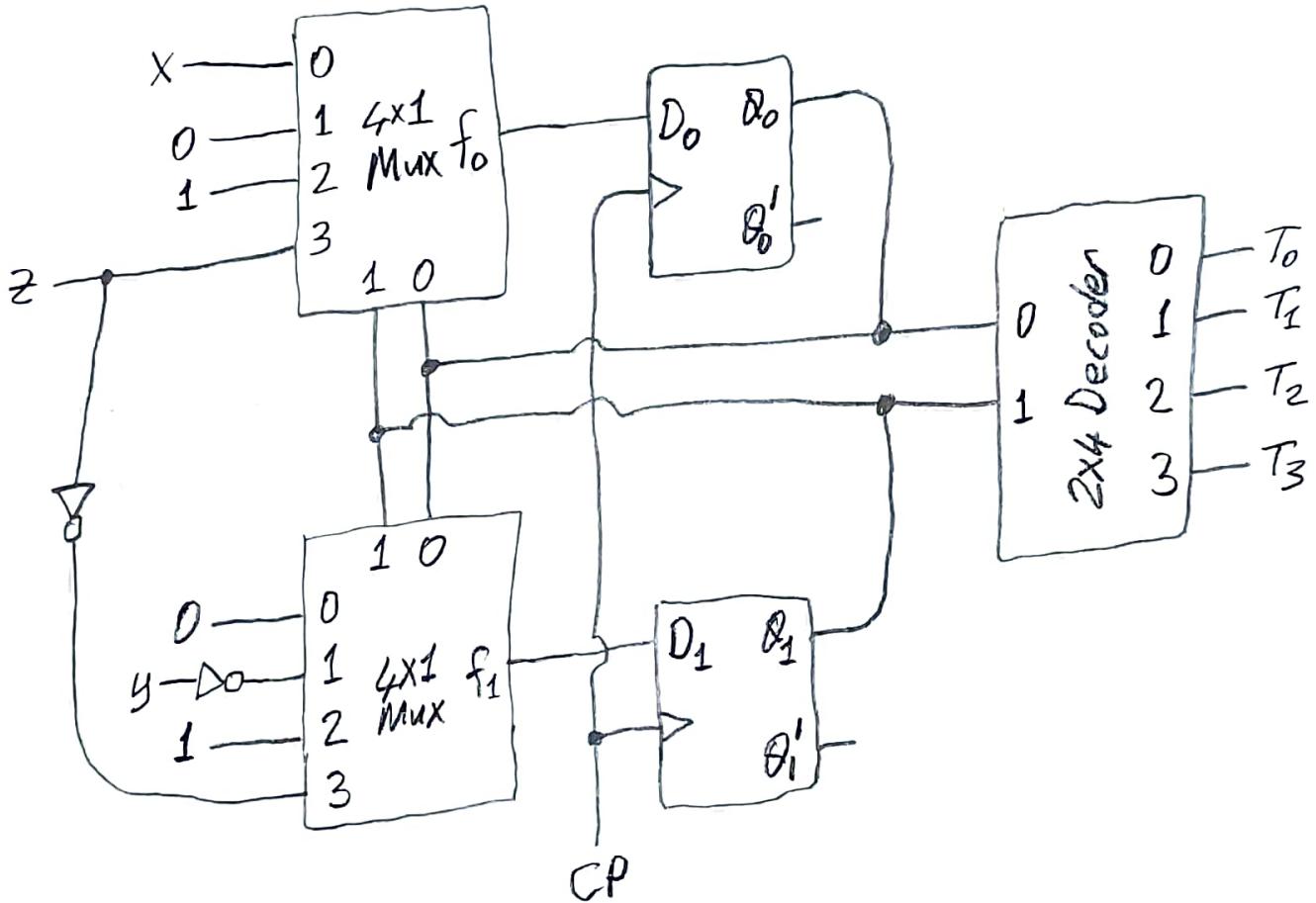


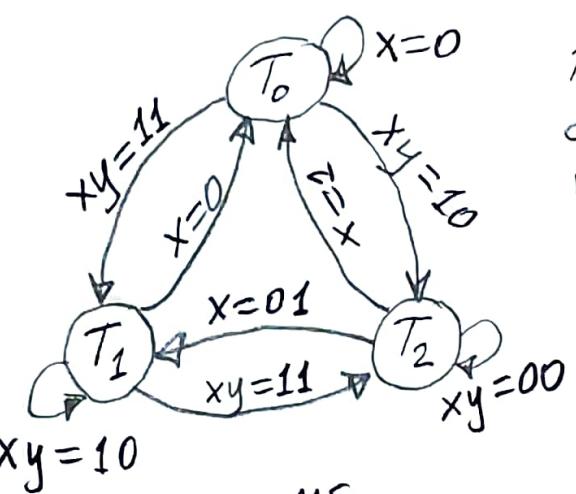


$$T_0 = 00 \quad T_2 = 10 \quad T_1 = 01 \quad T_3 = 11$$

Herhangi bir ASM devresinin durum diyagramı yukarıda verilmiştir. Kontrol devresini Multiplexer, Decoder, D flip-flop ve kopü elemanları kullanılarak tasarıla.

PS	NS	Girdi Sartları	Mux. Girişleri	
$D_1 D_0$	$D_1 D_0$		Mux1	Mux0
00	00	x'	0	x
00	01	x		
01	00	y	y'	0
01	10	y'		
10	11	1	1	1
11	01	z	z'	
11	10	z'		z





Herhangi bir ASM devresinin 42 durum diyagramı yandırı verilmiştir. Kontrol devresinin Multiplexer, Decoder, D flip-flop ve kapı elementleri kullanılarak tasarlayınız.

$$T_0 = 00, T_1 = 01, T_2 = 11$$

PS	NS	Girdi Sartları	Mux. Girişleri	
$Q_1 Q_0$	$Q_1 Q_0$		Mux1	Mux0
00	00	x'		
00	01	xy	xy'	x
00	11	xy'		
01	00	x'		
01	01	xy'	xy	x
01	11	xy		
11	00	x	$x'y'$	x'
11	01	$x'y$	$= (x+y)'$	
11	11	$x'y'$		

$T_2 = 11$
olmrsa
kontrol
devresinde
daha az
kapı elementi
kullanılır.

