1. SAYI SİSTEMLERİ

Günümüzde kullanılan sayı sistemi on tabanlı sayı sistemidir. Onlu sayılarla mikroişlemcilerin ya da mikro bilgisayarların verileri işlemesi zor ve karmaşıktır. Sayısal elektronik devre düzeneklerine gereksinim hızlı bir şekilde artmış ve araştırmalar sayısal (ikili) veri işleme doğrultusunda gerçekleştirilmiştir. İkili sayılar iki rakamdan oluşur, dolayısıyla iki seviye ile işlem yapılır. Onlu sayılarda ise on farklı rakamolduğundan on seviyeli bilgi ve veri işleme gerçekleştirilmelidir. Bu durum devre düzeneği tasarımı ve gerçekleştirilmesi açısından ikili sayıların onlu sayılara göre elektronikte tercih edilmesinin açık göstergesidir.

1.1.(10) Tabanlı (Decimal) Sayılar

Günlük hayatta kullanılan sayılar 10 tabanlı sayı sistemine aittir. 10 sayısı sistemde kullanılan rakam sayısını verir.
a={0,1,2,3,4,5,6,7,8,9}

1.2.(2) Tabanlı (Binary) Sayılar

Tabanı iki olan sayıların oluşturduğu bir sayı sistemidir. Sayı sistemleri içinde en sade sayı sistemidir.
a ={0,1}

Verilen iki tabanlı sayının on tabanlı karşılığını bulmak için her bit'i ağırlığına göre yazmak gerekir. Bit: İkili sayı sisteminde her bir basamağa verilen isimdir (Binary Digit).

İkili düzende en az ağırlıklı bit'in karşılığı LSB (Least Signification Bit) ve en çok ağırlıklı bit'in karşılığı MSB (Most Signification Bit)'dir.

$(11101101)_2$=1.27+1.26+1.25+0.24+1.23+1.22+0.21+1.20=128+64+32+0+8+4+0+1=$(237)_{10}$

$(1011001011)_2$=
$(1001,1011)_2$=

4 Bit = 1 Nibble
8 Bit = 1 Byte = 1B
16 Bit = 2 Byte = 1 Word = 1W
32 Bit = 4 Byte = 2 Word = 1 Long Word
1024 Byte = 1 Kilo Byte = 1KB
1024 KB = 1 Mega Byte = 1MB
1024 MB = 1 Giga Byte = 1GB
1024 GB = 1 Tera Byte = 1TB

1.3.Sekizli (Octal) Sayılar

Tabanı sekiz olan sayıların oluşturduğu bir sayı sistemidir. Bu sayı sisteminin rakam kümesi;
a = {0,1,2,3,4,5,6,7}

$(276)_8$ = X2.82+ X1.81+X0.80=2.82+7.81+6.80=128+56+6=$(190)_{10}$
$(42105)_8$= 4.84+2.83+1.82+0.81+5.80=16384+1024+64+0+5=$(17477)_{10}$
$(7652)_8$=
$(243,76)_8$=

1.4.Onaltılı (Hexadecimal) Sayılar

Tabanı on altı olan sayıların oluşturduğu bir sayı sistemidir. Hexadecimal sayı sisteminde 10 tabanlı sayılarda olan rakamlara ilave olarak altı rakam daha vardır. (A=10, B=11, C=12, D=13, E=14 ve F=15)
a ={0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

$(10BA)_{16}$ = X3.163+X2.162+X1.161+X0.160=1.163+0.162+B.161+A.160=4096+0+176+10=$(4282)_{10}$
$(CE2)_{16}$= C.162+E.161+2.160=3072+224+2=$(3298)_{10}$
$(2A4F)_{16}$=
$(FA9,12)_{16}$=

1.5.On Tabanlı Sayı Sisteminden İki Tabanlı Sayı Sistemine Dönüşüm

Verilen sayı tam sayı ise ikiye bölünerek kalanlar kayıt edilir. Bu durumda kalan kısım ya 0 ya da 1'dir. Bölme işlemi bölümde 0 veya 1 görülünceye kadar devam ettirilir.

$(248)_{10}$ = (..?..)$_2$=$(11111000)_2$

Bölüm	Kalan	LSB
248 / 2 = 124	0	
124 / 2 = 62	0	
62 / 2 = 31	0	
31 / 2 = 15	1	
15 / 2 = 7	1	
7 / 2 = 3	1	
3 / 2 = 1	1	
	MSB	

$(0,625)_{10}$ = (..?..)$_2$=$(0,101)_2$

Çarpım	Tam Kısım	
0,625 * 2 = 1,250	1	MSB
0,250 * 2 = 0,500	0	
0,5 * 2 = 1,000	1	LSB

1.6.On Tabanlı Sayı Sisteminden Sekiz veya On altı Tabanlı Sayı Sistemine Dönüşüm

On tabanlı sayıdan, sekiz tabanı sayı veya on altı tabanlı sayıya geçişte onlu sayıdan ikiliye geçişte yapıldığı gibi tam kısım için taban sayısına bölme, kesirli kısım için taban sayısı ile çarpma işlemi yapılır.
$(247)_{10}$=$(367)_8$
$(0,513)_{10}$= (…….)$_8$
$(1367)_{10}$=(……)$_{16}$

1.7.İki Tabanlı Sayı Sisteminden Sekiz Tabanlı Sayı Sistemine Dönüşüm

Tabanı sekiz olan sayıların oluşturduğu sayı kümesindeki tüm sayılar {0,1,2,3,5,6,7}, iki tabanlı sayı sisteminde **üç bit** ile ifade edilmektedir {000, 001, 010, 011, 100, 101, 110, 111}.
İki tabanlı sayı sisteminde verilen sayı virgülden sağa ve sola doğru olmak üzere üçer bitlik parçalara ayrılır. Son parçalar üç bit olmuyorsa sayısal değeri bozmayacak şekilde sıfır eklenir ve üç bite tamamlanır. Bu üç bitlik grupların sekiz tabanlı sayı sisteminde karşılığı yazılarak verilen sayı iki tabanlı sayı sisteminden sekiz tabanlı sayı sistemine çevrilmiş olur.

$(1111110,01011)_2$= 00(1 111 110 , 010 11)0_2=$(176,26)_8$
$(101000111,00100011110)_2$= (…….)$_8$
$(1110011100,10011011)_2$= (…….)$_8$

1.8.Sekizli Tabanlı Sayı Sisteminden İki Tabanlı Sayı Sistemine Dönüşüm

Daha önce anlatılan, iki tabanlı sayı sisteminden sekiz tabanlı sayı sistemine dönüşüm işleminin tam tersi yapılır. Verilen sekiz tabanlı sayıya ait rakamların ikili düzende üç bit olarak karşılığı yazılıp sayısal değer bozulmayacak şekilde yan yana sıralanırsa, dönüşüm işlemi doğrudan gerçekleşmiş olmaktadır.
$(176,26)_8$=(001 111 110 , 010 110)$_2$=$(1111110,01011)_2$
$(451,3045)_8$=(……..)$_2$
$(71023,76)_8$=(……..)$_2$

1.9.İki Tabanlı Sayı Sisteminden On Altı Tabanlı Sayı Sistemine Dönüşüm

Tabanı on altı olan sayıların oluşturduğu sayı kümesindeki tüm sayılar {0,1,2,3,5,6,7,8,9,A,B,C,D,E,F,}, iki tabanlı sayı sisteminde **dört bit** ile ifade edilmektedir {0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111}. İki tabanlı sayı sisteminde verilen sayı virgülden sağa ve sola doğru olmak üzere dörder bitlik parçalara ayrılır. Son parçalar dört bit olmuyorsa sayısal değeri bozmayacak şekilde sıfır eklenir ve dört bite tamamlanır. Bu dört bitlik grupların on altı tabanlı sayı sisteminde karşılığı yazılarak verilen sayı iki tabanlı sayı sisteminden son altı tabanlı sayı sistemine çevrilmiş olur.

$(1111110,01011)_2=0(111\ 1110\ ,\ 0101\ 1)000_2=(7E,58)_{16}$
$(1011000100,1010011)_2=(\ldots\ldots\ldots)_{16}$
$(1001001111001,101001110)_2=(\ldots\ldots\ldots)_{16}$

1.10. Onaltı Tabanlı Sayı Sisteminden İki Tabanlı Sayı Sistemine ve Diğer Tabanlara Dönüşüm

Daha önce anlatılan, iki tabanlı sayı sisteminden on altı tabanlı sayı sistemine dönüşüm işleminin tam tersi yapılır. Verilen on altı tabanlı sayıya ait rakamların ikili düzende dört bit olarak karşılığı yazılıp sayısal değer bozulmayacak şekilde yan yana sıralanırsa, dönüşüm işlemi doğrudan gerçekleşmiş olmaktadır. Onaltılık tabandan diğer tabanlara dönüşümde önce onluk tabana sonra istenilen tabana dönüştürülür.

$(34EA,B28)_{16}=(0011\ 0100\ 1110\ 1010\ ,\ 1011\ 0010\ 1000)_2=(11010011101010,101100101)_2$
$(1C94,2F9)_{16}=(\ldots\ldots\ldots)_2$
$(A53B,762)_{16}=(\ldots\ldots\ldots)_2$
$(4A8)_{16}=(\ldots\ldots\ldots)_5$

1.11. Tabanlarda Dört işlem

$(244)_5+(332)_5=$
$(4A8)_{16}+(BC6)_{16}=$
$(11001000)_2+(01101010)_2=$

2. SAYILARIN KODLANMASI

Hayatımızda kullanılan onlu sistemdeki sayılar, özel karakter ve harfler, dijital sistemlerdeişlenebilmesi için ikili sayı sistemine dönüştürülmesi gerekir. Bilgileri dijital sistemlerdekullanmak ve üzerinde işlem yapmak için yapılan dönüştürme işlemine kodlama adı verilir. Başka bir deyişle kodlama iki küme arasında karşılığı tanımlanmış temel kurallar dizini olarakda tanımlanır. Kodlar kendi arasında dijital ve alfanumerik olmak üzere iki temel türdeincelenebilir.

Kodlama Çeşitleri

- Dijital (sayısal) Kodlama
  - i) BCD (Binary Coded Decimal - İkili Kodlanmış Onlu Sayı Kodu) veya 8421 Kodlama
  - ii) Gray Kodu
  - iii) Artı-3 (Excess-3) Kodu
  - iv) 5'te 2 Kodu
  - v) Eşlik (Parity) Kodu
  - vi) Aiken Kodu
  - vii) Bar (Çubuk) Kodu (Bar-Code)
- Alfanumerik Kodlama
  - i) ASCII Kodu

2.1.BCD Kodlama

Binary Coded Decimal olarak bilinen bu kod on tabanlı sayıları ikili düzende kodlanmasıdemektir. Bu kodlamada, on tabanlı sayının her bir basamağı (digit) için dörder bitlik ikili birifade yazılarak kodlama yapılır.

On Tabanlı Sayı	BCD Kodlama (8421)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Yukarıdaki tablodan da görüldüğü gibi onluk sayı sistemi 0 ile 9 arasındaki sayılarıiçerdiğinden, her basamaktaki sayının ikili sistemde kodlanması için 4 bite ihtiyaç vardır. Onlubir sayıyı BCD kodlu olarak yazmak için onlu sayının her bir basamağı 4 bitlik iki tabanlı sayıgrupları şeklinde yazılır. Yazılan gruplar bir araya getirilince BCD kodlu sayı elde edilir.

$(145)_{10}=(0001\ 0100\ 0101)_{BCD}$

$(5698)_{10}=(\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..)_{BCD}$

$(73204)_{10}=(\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..)_{BCD}$

$(1001011010000100)_{BCD}=(1001\ 0110\ 1000\ 0100)_{BCD}=(\ldots\ldots\ldots\ldots\ldots)_{10}$

$(0111001010010110)_{BCD}=(\ldots\ldots..\ \ldots\ldots..\ \ldots\ldots..\ \ldots\ldots..)_{BCD}=(\ldots\ldots\ldots\ldots\ldots..)_{10}$

$(100001110011010110010011)_{BCD}=(\ldots\ldots..\ \ldots\ldots..\ \ldots\ldots..\ \ldots\ldots..\ \ldots\ldots..\ldots\ldots..)_{BCD}=(\ldots\ldots\ldots\ldots\ldots..)_{10}$

2.2.GRAY Kodu

Dijital elektronik ve bilgisayar giriş çıkış işlemlerinde kullanılan Gray kodlama yöntemi, en azdeğişim kodlamadır. Bunun nedeni bir sayıdan diğerine geçerken yalnızca bir bitin konumdeğiştirmesidir. Örneğin; ikili (binary) kodlamada $(3)_{10}= (0011)_2$ değerinden $(4)_{10} = (0100)_2$değerine geçerken üç bitin değeri aynı anda değişirken, gray kodlamada yalnızca bir bitindeğeri değişir.

İkili sayı sistemine kolayca çevrilmesi avantajıdır. En çok tercih edilen uygulama alanı olarakgeri beslemeli sistemlerde konum denetimidir. Kodlama sıralamasında bir önceki sayısal kodile bir sonraki sayısal kod arasında sadece tek bir bit'de farklılık olmasından dolayı konumbelirleme işlemlerinde tercih edilir. Gray kodlamada basamakların sayı değeri yoktur.

On Tabanlı Sayı	İki Tabanlı Sayı	Gray Kodu
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010

2.2.1. İkili Sayılardan Gray Koduna Geçiş

İkili sayı sisteminden gray koduna geçerken en ağırlıklı bit (MSB) hangi sayı olursa olsundoğrudan yazılır. Daha sonra her bit solundaki bit ile (bir üst bit ile bir alt bit) ardışıktoplanarak bir sonraki basamağa yazılır. İşlem esnasında oluşan eldeler atılır. Bu işleme en azağırlıklı bite kadar (LSB) devam edilir.

$(1\ 1\ 1\ 1)_2=(1\ 0\ 0\ 0)_{gray}$

$(1000)_2=(\ldots\ldots\ldots\ldots..)_{gray}$

$(0111)_2=(\ldots\ldots\ldots\ldots..)_{gray}$

2.2.2. Gray Kodundan İkili Sayı Sistemine Geçiş

Gray kodlu bir sayıyı ikili sistemdeki sayı şekline dönüştürmek için, en ağırlıklı bit (MSB)doğrudan aşağı yazılır. Aşağı yazılan sayı ile bir sonraki basamakta bulunan sayı toplanarakönceki yazılan sayının yanına yazılır. Bu işleme en düşük değerlikli bite kadar devam edilir. İşlem esnasında oluşan eldeler atılır.

$(1\ 1\ 1\ 0)_{gray}$

$(1\ 0\ 1\ 1)_2$

$(1000)_{gray}=(\ldots\ldots\ldots\ldots..)_2$

$(0111)_{gray}=(\ldots\ldots\ldots\ldots..)_2$

$(1101)_{gray}=(\ldots\ldots\ldots\ldots..)_{10}$

2.3.Artı-3 (Excess-3) Kodu

On tabanlı sayıya 3 eklenip BDC koda çevrilmesiyle, Artı-3 kodlama elde edilir. Aritmetikişlemlerde işlem kolaylığı sağladığı için kullanılır. İki veya daha fazla basamaklı sayılar +3koduna dönüştürülürken her basamak için aynı işlem yapılır.

$(7)_{10}=(1010)_{+3}$

$(24)_{10}=(0101\ 0111)_{+3}$

$(138)_{10}=(0100\ 0110\ 1011)_{+3}$

$(4296)_{10}=(\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..)_{+3}$

$(25792)_{10}=(\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..)_{+3}$

+3 kodunda on tabanlı sayıya dönüşüm için yukarıda anlatılan işlemin tersi yapılır.

$(1001011010000100)_{+3}=(1001\ 0110\ 1000\ 0100)_{+3}=(9\ 6\ 8\ 4)=(6351)_{10}$

$(0111001110010110)_{+3}=(\ldots\ldots..\ \ldots\ldots..\ \ldots\ldots..\ \ldots\ldots..)_{+3}=(\ldots\ldots\ldots\ldots\ldots\ldots)=(\ldots\ldots\ldots\ldots..)_{10}$

$(10000111001101011001 0011)_{+3}=(\ldots\ldots..\ \ldots\ldots..\ \ldots\ldots..\ \ldots\ldots..\ \ldots\ldots..$
$\ldots\ldots..)_{+3}=(\ldots\ldots\ldots)=(\ldots\ldots\ldots\ldots..)_{10}$

2.4.(5'te 2) Kodu

5'te 2 kodunda on tabanındaki her rakam beş bitlik 2 tabanlı sayı ile ifade edilmektedir. 5'te 2 koduyla kodlanmış her rakamın içinde sadece 2 adet "1" bulunmaktadır. 5' te 2 kodlanmışsayıların basamak ağırlıkları (7 4 2 1 0) ' dır. (0)10 rakamı 5' te 2 kodunda (11000) şeklindeifade edilir.

On Tabanlı Sayı	5' te 2 Kodu
0	11000
1	00011
2	00101
3	00110
4	01001
5	01010
6	01100
7	10001
8	10010
9	10100

$(5)_{10}=(01010)_{5'te2}$

$(38)_{10}$=(00110 10010) $_{5'te2}$
$(297)_{10}$=(00101 10100 10001) $_{5'te2}$
$(5640)_{10}$=(……..)$_{5'te2}$
$(10856)_{10}$=(……..)$_{5'te2}$
$(100100110000101)_{5'te2}$=(10010 01100 00101)$_{5'te2}$=$(862)_{10}$
$(10100101001010001100)_{5'te2}$=(…….. …….. …….. …….. ……..)$_{5'te2}$=(……)$_{10}$

2.5.Eşlik Kodu (Hata Sezici Kodlama)

Dijital bilgilerin iletimi esnasında oluşabilecek hataların belirlenmesinde kullanılanyöntemdir. Bu yöntem ile hata düzeltilmez sadece varlığı tespit edilir. Bu yöntemdekodlanmış sayının sağına veya soluna bir eşlik (parity) biti eklenir. Eşlik biti kodlanan veride 0veya 1'lerin tek veya çift olduğunu belirtir. İki farklı eşlik biti yöntemi vardır. Bunlar, çift eşlik(even parity) ve tek eşlik (odd parity) yöntemidir.

(1001001)2=(11001001)EP1
(1001001)2=(01001001)OP1

2.6.Aiken Kodu

Aiken kodunda on tabanındaki her rakam dört bitlik 2 tabanlı sayı ile ifade edilmektedir.Aiken kodu ile kodlanmış sayıların basamak ağırlıkları (2 4 2 1) ' dir. 5' ten küçük olanrakamlarda en ağırlıklı bit (MSB) sıfır, 5'eşit ve daha büyük rakamlarda en ağırlıklı bit bir' dir.

On Tabanlı Sayı	Aiken Kodu
0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

Aiken kodlama simetrik kodlamaya iyi bir örnektir. Bu kodlamada (0-4) arasındaki rakamlarbulunurken BCD kodlama yapılır. (5-9) arasındaki rakamlar bulunurken ise (0-4) arasındakirakamların Aiken kodu karşılığının tümleyeni alınır. Örneğin 5'in Aiken kodu, 4'ün Aiken kodkarşılığının tümleyenidir.

(3)10=(0011)Aiken
(21)10=(0010 0001) Aiken
(297)10=(0010 1111 1101)Aiken
(5640)10=(……..)Aiken
(13856)10=(……..)Aiken

2.7.Bar Kod

Bar Kodlamada veriler, farklı kalınlıktaki paralel çizgiler ve boşluklar ile kodlanır. Barkodların en iyibilinen ve en yaygın kullanımı tüketici ürünlerindedir.

2.8.ASCII Kodu

ASCII (American Standard Code for Information Interchange) (Bilgi Değişimi İçin Amerikan Standart Kodlama Sistemi). Latin alfabesi üzerine kurulu 7 bitlik bir karakter setidir. İlkkez 1963 yılında ANSI tarafından standart olarak sunulmuştur.

ASCII'de 33 tane basılmayan kontrol karakteri ve 95 tane basılan karakter bulunur. Kontrolkarakterleri metnin akışını kontrol eden, ekranda çıkmayan karakterlerdir. Basılan karakterlerise ekranda görünen, okuduğumuz metni oluşturan karakterlerdir. ASCII'nin basılankarakterleri aşağıda belirtilmiştir. Bütün büyük ve küçük harfler, rakamlar, noktalamaişaretleri ve kontrol karakterleri bu kodlamada tanımlanmıştır.

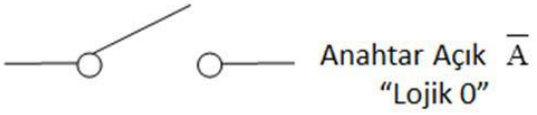
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	|	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	—	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	n
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	2
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

3. LOJİK KAPILAR (LOGIC GATES)

Dijital (Sayısal) devrelerin tasarımında kullanılan temel devre elemanlarına Lojik kapılar adıverilmektedir. Her lojik kapının bir çıkışı, bir veya birden fazla girişi vardır. Lojik kapılarıngirişlerine, "Lojik 1" veya "Lojik 0" adı verilen seviyeler uygulanabilir. Girişlerinin durumunagöre lojik kapıların çıkışından "Lojik 1" veya "Lojik 0" gerilim seviyeleri gözlemlenir. "Lojik 0"seviye, 0 (sıfır) volt gerilimi temsil etmekte iken (Lojik 0 ≡ 0V). "Lojik 1" seviye ise, +5 voltgerilimi temsil etmektedir (Lojik 1 ≡ 5V).

Lojik kapıların girişlerine giriş seviyesinin uygulanması ("Lojik 1" ve "Lojik 0") için iki konumludevre elemanı olan anahtar kullanılır. Anahtarın iki konumu vardır (anahtar açık ve anahtarkapalı). Genelde, anahtar açık durumu "Lojik 0", anahtar kapalı durumu ise "Lojik 1" olaraktemsil edilmektedir.

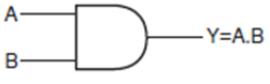


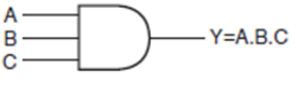
VE (And), VEYA (Or), DEĞİL (Not) olmak üzere üç temel lojik kapı vardır. Ayrıca bu üç temelkapıdan türetilmiş (VE-DEĞİL (Nand), VEYA-DEĞİL (Nor), ÖZEL VEYA (Ex-Or) ve ÖZEL VEYADEĞİL(Ex-Nor)) dört kapı ile birlikte toplamda 7 (yedi) adet lojik kapı bulunmaktadır.

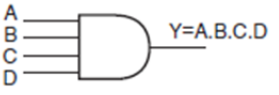
Bir Lojik kapının girişlerinin durumlarına bağlı olarak çıkışının ne olacağını gösteren tabloyadoğruluk tablosu (truth table) adı verilir. Doğruluk tablosu n girişli bir lojik kapının, olası tüm giriş durumuna karşılık, lojik kapının çıkışının hangi değeri alacağını gösterir. n girişli bir lojikkapının, girişlerinin alabileceği 2n adet durum vardır ve her durum doğruluk tablosundabulunmalıdır.

3.1.VE KAPISI (AND GATE)

Aşağıda 2, 3 ve 4 girişli VE kapılarının sembolleri, lojik ifadeleri ve 2 girişli VE kapısına aitdoğruluk tablosu görülmektedir. Doğruluk tablosu incelendiğinde, 2 girişli VE kapısının 2^2=4durumu olduğu görülmektedir. Doğruluk tablosundan da görüldüğü üzere VE kapısının her ikigirişi "lojik 1" durumda iken çıkışı "lojik 1" olmaktadır.

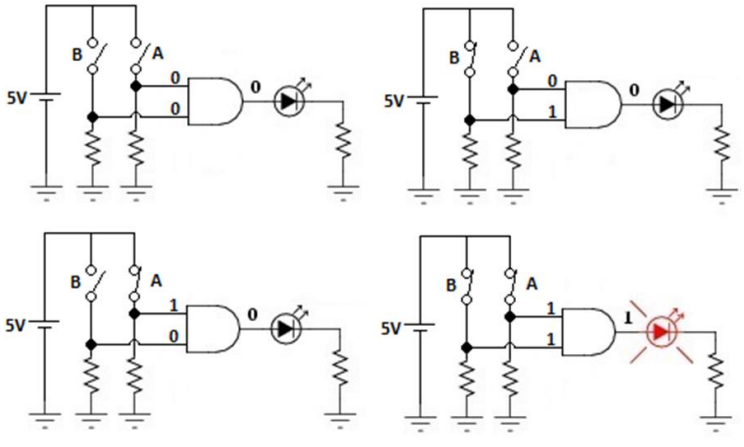




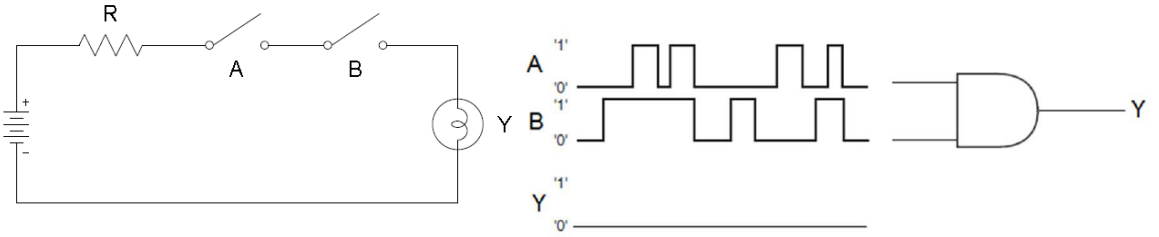


A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

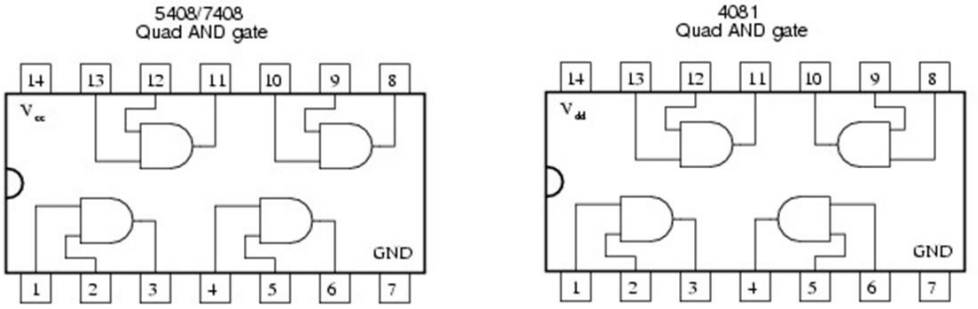
2 girişli VE kapısına ait doğruluk tablosunun nasıl oluşturulduğu aşağıdaki şekillerde ayrıntılıolarak görülmektedir.



Aşağıda VE kapısının elektriksel eşdeğer devresi verilmiştir. VE kapısı matematiksel anlamdabir bitlik çarpma işlemini ifade ederken elektriksel anlamda anahtarların seri bağlanmasıniifade etmektedir. Aşağıdaki devrede, Y lambasının yanıyor olması “lojik 1”, Y lambasınınsönük olması ise “lojik 0” anlamına gelmektedir. Şekilden de görüleceği üzere Y lambasınınyanması için her iki anahtarın (A ve B) kapalı olması gerekmektedir.

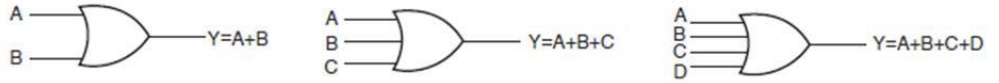


Aşağıda TTL teknolojisi ile üretilmiş 2 girişli VE kapı entegresinin (7408, 5408) ve CMOSteknolojisi ile üretilmiş 2 girişli VE kapı entegresinin (4081) iç yapısı verilmiştir.



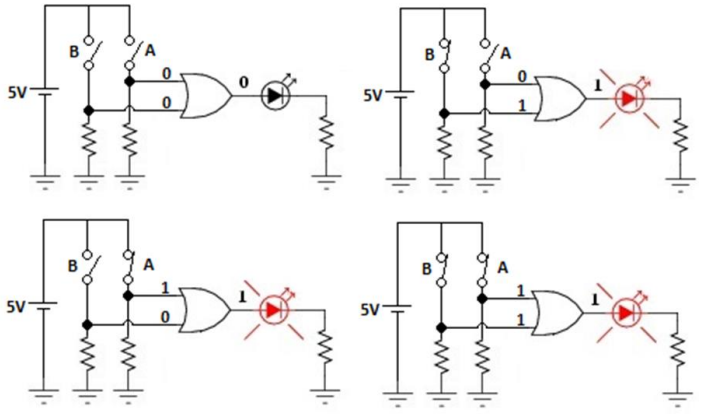
3.2.VEYA KAPISI (OR GATE)

Aşağıda 2, 3 ve 4 girişli VEYA kapılarının sembolleri, lojik ifadeleri ve 2 girişli VEYA kapısına aitdoğruluk tablosu görülmektedir. Doğruluk tablosundan da görüldüğü üzere VEYA kapısınınherhangi bir girişi “lojik 1” veya her iki girişi de “lojik 1” durumda iken çıkışı “lojik 1”olmaktadır.

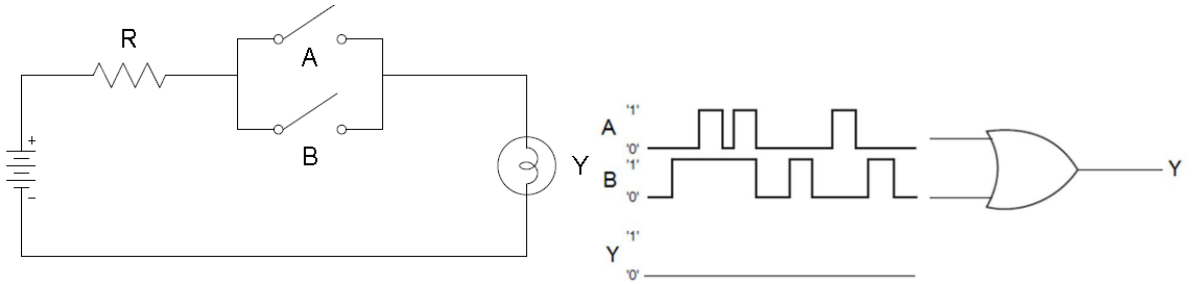


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

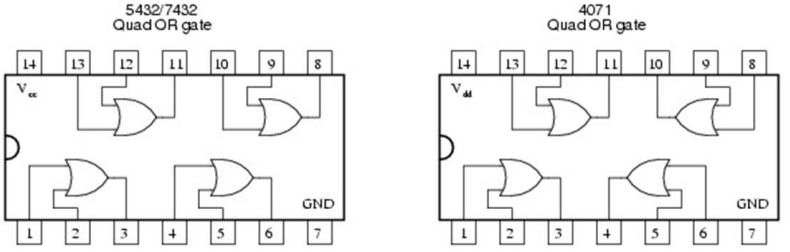
2 girişli VEYA kapısına ait doğruluk tablosunun nasıl oluşturulduğu aşağıdaki şekillerdeayrıntılı olarak görülmektedir.



Aşağıda VEYA kapısının elektriksel eşdeğer devresi verilmiştir. VEYA kapısı elektrikselanlamda anahtarların paralel bağlanmasını ifade etmektedir. Aşağıdaki devrede, Ylambasının yanıyor olması “lojik 1”, Y lambasının sönük olması ise “lojik 0” anlamınagelmektedir. Şekilden de görüleceği üzere Y lambasının yanması için anahtarlardan herhangibirinin (A ve B) veya ikisinin de kapalı olması gerekmektedir.

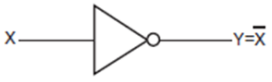


Aşağıda TTL teknolojisi ile üretilmiş 2 girişli VEYA kapı entegresinin (7432, 5432) ve CMOSteknolojisi ile üretilmiş 2 girişli VEYA kapı entegresinin (4071) iç yapısı verilmiştir.



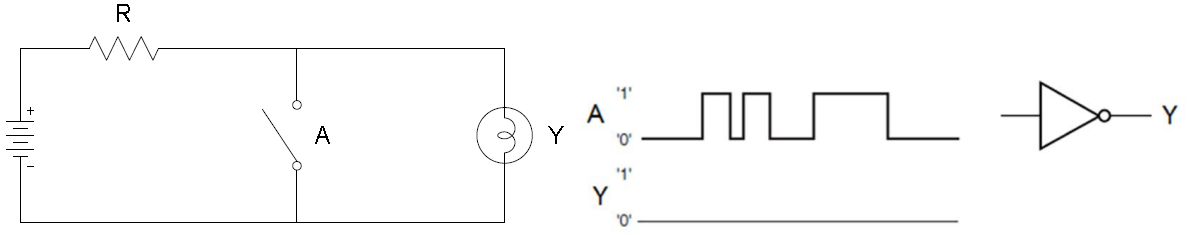
3.3.DEĞİL KAPISI (NOT GATE)

Aşağıda DEĞİL kapısının sembolü, lojik ifadesi ve doğruluk tablosu görülmektedir. Doğruluktablosundan da görüldüğü üzere DEĞİL kapısının çıkışı, girişinin evriğidir (tümleyenidir).

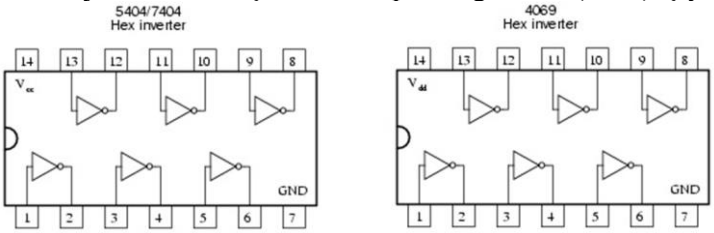


X	Y
0	1
1	0

Aşağıda DEĞİL kapısının elektriksel eşdeğer devresi verilmiştir. DEĞİL kapısı veya DEĞİL işlemi(tümleyen alma işlemi) elektriksel anlamda anahtarın ve/veya anahtarların çıkışa (veya çıkışabağlı lambaya) paralel bağlanmasını ifade etmektedir. Şekilden de görüleceği üzere Ylambasının yanması için A anahtarının açık olması gerekmektedir. A anahtarı kapalı olduğuzaman çıkış veya çıkışa bağlı lamba anahtar üzerinden kısa devre olmakta ve yanmamaktadır.

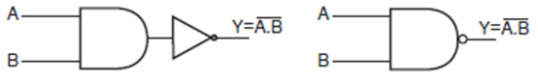


Aşağıda TTL teknolojisi ile üretilmiş DEĞİL kapı entegresinin (7404, 5404) ve CMOS teknolojisiile üretilmiş DEĞİL kapı entegresinin (4069) iç yapısı verilmiştir.



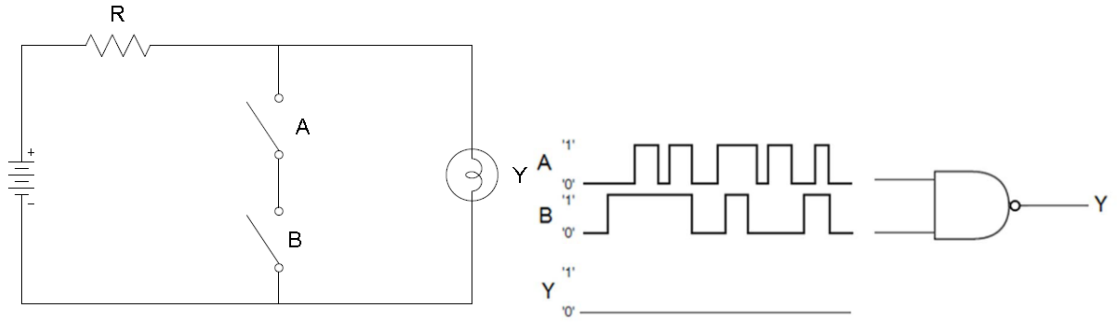
3.4.VE-DEĞİL KAPISI (NAND GATE)

Aşağıda 2 girişli VE-DEĞİL kapısının sembolü, lojik ifadesi ve 2 girişli VE-DEĞİL kapısına aitdoğruluk tablosu görülmektedir.

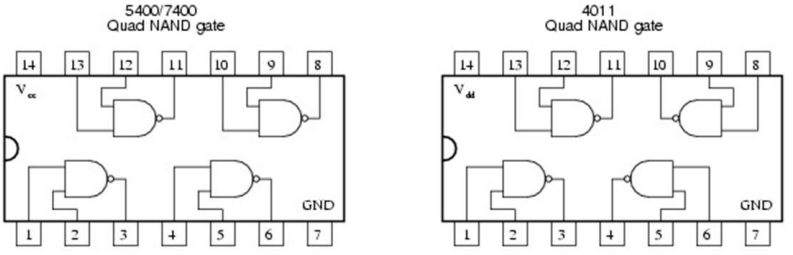


A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Aşağıda VE-DEĞİL kapısının elektriksel eşdeğer devresi verilmiştir.

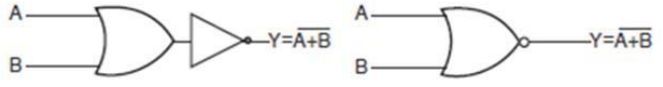


Aşağıda TTL teknolojisi ile üretilmiş 2 girişli VE-DEĞİL kapı entegresinin (7400, 5400) ve CMOSteknolojisi ile üretilmiş 2 girişli VE-DEĞİL kapı entegresinin (4011) iç yapısı verilmiştir.



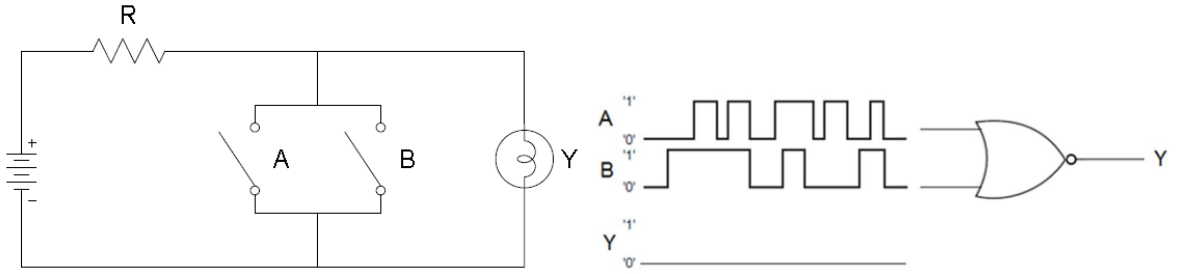
3.5.VEYA-DEĞİL KAPISI (NOR GATE)

Aşağıda 2 girişli VEYA-DEĞİL kapısının sembolü, lojik ifadesi ve 2 girişli VEYA-DEĞİL kapısınaait doğruluk tablosu görülmektedir.

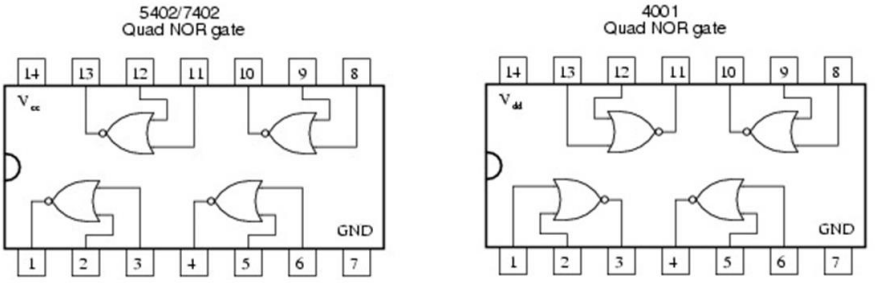


A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Aşağıda VEYA-DEĞİL kapısının elektriksel eşdeğer devresi verilmiştir.

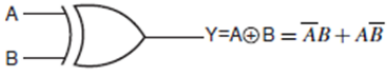


Aşağıda TTL teknolojisi ile üretilmiş 2 girişli VEYA-DEĞİL kapı entegresinin (7402, 5402) veCMOS teknolojisi ile üretilmiş 2 girişli VEYA-DEĞİL kapı entegresinin (4001) iç yapısıverilmiştir.



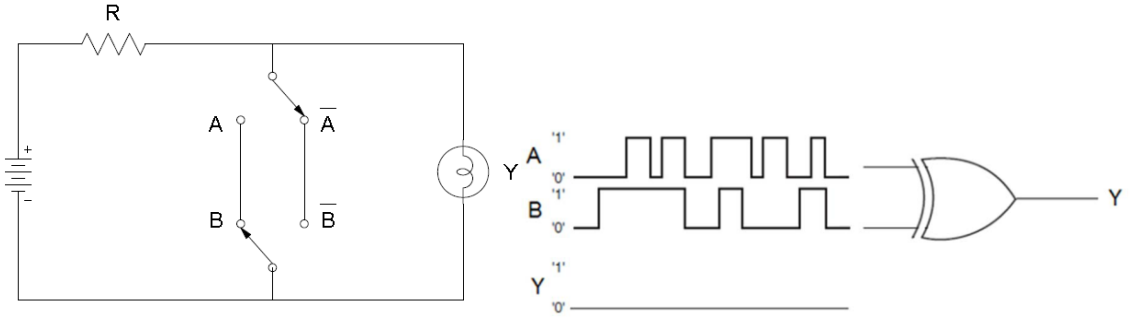
3.6.ÖZEL VEYA KAPISI (EX-OR GATE)

Aşağıda 2 girişli ÖZEL VEYA kapısının sembolü, lojik ifadesi ve 2 girişli ÖZEL VEYA kapısına aitdoğruluk tablosu görülmektedir.



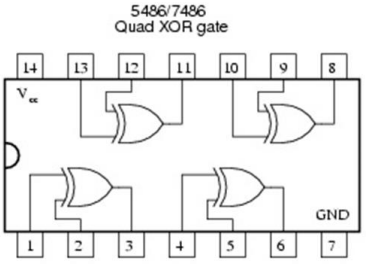
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

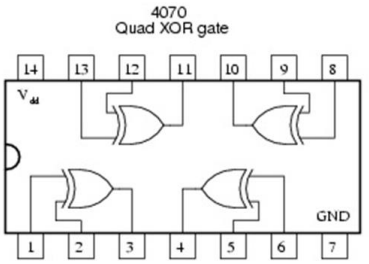
Aşağıda ÖZEL VEYA kapısının elektriksel eşdeğer devresi verilmiştir.



Doğruluk tablosundan da görüldüğü gibi kapı girişleri bir bit olarak düşünüldüğünde,devrenin çıkışına yansıyan Y lambasının yanma işlevi bir bit eşitsizlik devresi olarak da ifadeedilir.

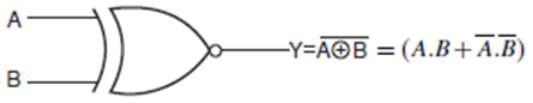
Aşağıda TTL teknolojisi ile üretilmiş 2 girişli ÖZEL VEYA kapı entegresinin (7486, 5486) veCMOS teknolojisi ile üretilmiş 2 girişli ÖZEL VEYA kapı entegresinin (4070) iç yapısı verilmiştir.





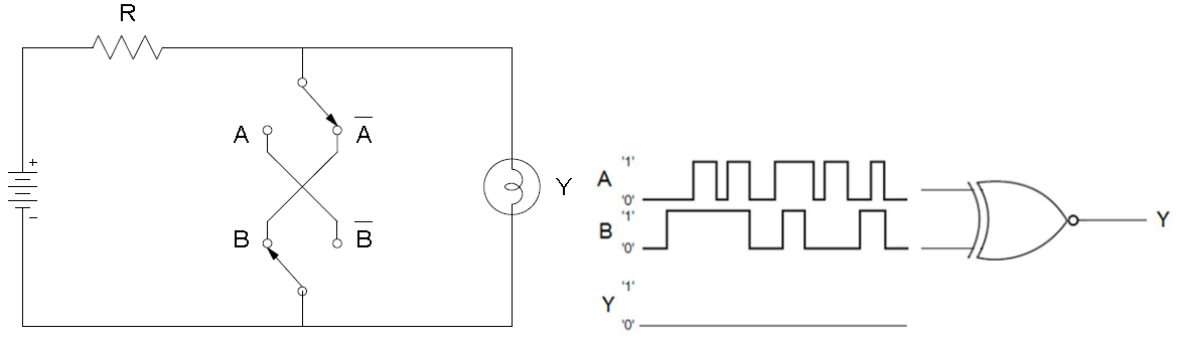
3.7.ÖZEL VEYA-DEĞİL KAPISI (EX-NOR GATE)

Aşağıda 2 girişli ÖZEL VEYA-DEĞİL kapısının sembolü, lojik ifadesi ve 2 girişli ÖZEL VEYA-DEĞİLkapısına ait doğruluk tablosu görülmektedir.



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Aşağıda ÖZEL VEYA-DEĞİL kapısının elektriksel eşdeğer devresi verilmiştir.

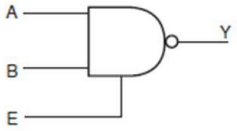


Doğruluk tablosundan da görüldüğü gibi kapı girişleri bir bit olarak düşünüldüğünde,devrenin çıkışına yansıyan Y lambasının yanma işlevi bir bit eşitlik devresi olarak da ifadeedilir.

3.8.ÜÇ KONUMLU LOJİK KAPILAR (TRISTATE LOGIC GATES)

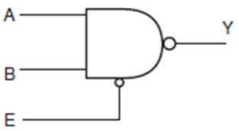
Bu konuya kadar anlatılan yedi lojik kapının “lojik 0” ve “lojik 1” olmak üzere iki çıkış durumu(seviyesi) bulunmaktadır. Ancak üç konumlu lojik kapıların, “lojik 0”, “lojik 1” çıkış durumları yanında Yüksek-Direnç çıkış durumu bulunmaktadır. Yüksek Direnç durumu lojik kapıda ekstra olarak bulunan Enable girişi ile kontrol edilmektedir. Bu konuya kadar anlatılan tüm kapılar (VE, VEYA, DEĞİL, VE-DEĞİL, VEYA-DEĞİL, ÖZEL VEYA, ÖZEL VEYA-DEĞİL) üç konumlu olarak üretilebilmektedir.

Kapılarda bulunan Enable girişi, “lojik 0” aktif veya “lojik 1” aktif olabilmektedir. Aşağıda Enable girişi “lojik 0” aktif ve “lojik 1” aktif olan 2 girişli VE-DEĞİL
(NAND) kapısının sembolleri ve doğruluk tabloları ayrı ayrı olarak verilmiştir.



A	B	E	Y
0	0	0	Y.D.
0	0	1	1
0	1	0	Y.D.
0	1	1	1
1	0	0	Y.D.
1	0	1	1
1	1	0	Y.D.
1	1	1	0

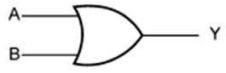
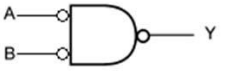
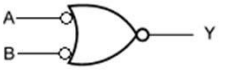
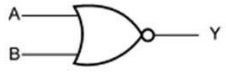
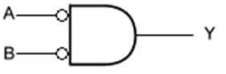
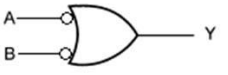
Y.D. = Yüksek Direnç Durumu



A	B	E	Y
0	0	0	1
0	0	1	Y.D.
0	1	0	1
0	1	1	Y.D.
1	0	0	1
1	0	1	Y.D.
1	1	0	0
1	1	1	Y.D.

Y.D. = Yüksek Direnç Durumu

3.9.Lojik Kapılar Arasındaki İlişkiler

	Lojik Kapı	Lojik İfadesi ve Eşdeğer İfadesi	Eşdeğer Lojik Kapı
OR		$Y = A + B = \overline{\overline{A} \cdot \overline{B}}$	
AND		$Y = A \cdot B = \overline{\overline{A} + \overline{B}}$	
NOR		$Y = \overline{A + B} = \overline{A} \cdot \overline{B}$	
NAND		$Y = \overline{A \cdot B} = \overline{A} + \overline{B}$	

Soru: Y = $\overline{(A.B + C.D)}$ ifadesini lojik kapılar kullanarak çiziniz?

Soru: Y = $\overline{(A + B) \cdot (C + D)}$ ifadesini lojik kapılar kullanarak çiziniz?

Soru: İki girişli VE-DEĞİL (NAND) kapıları kullanarak iki girişli VEYA (OR) kapısı elde ediniz?

Soru: İki girişli VEYA-DEĞİL (NOR) kapıları kullanarak iki girişli VEYA (OR) kapısı elde ediniz?

Soru: İki girişli VEYA-DEĞİL (NOR) kapıları kullanarak iki girişli VE (AND) kapısı elde ediniz?

Soru: $Y = \overline{(A.B + C.D)}$ ifadesini lojik kapılar kullanarak çiziniz?

Soru: $Y = \overline{(A + B).(C + D)}$ ifadesini lojik kapılar kullanarak çiziniz?

3.10. BOOLE CEBRİ (BOOLEAN ALGEBRA)

1850'li yıllarda George Boole tarafından geliştirilen Boole Cebri, sayısal devrelerin analiz ve tasarımını sağlayan matematiksel teoridir. Sayısal bilgisayar devreleri uygulamasında, ikili değişkenler üzerine tanımlanan sayısal (dijital) operasyonları gösterir. Boole Cebri ikili sayı sistemine dayanır. Bu sistemde yer alan '0' lar kapalı (off), yanlış (false)gibi ifadeleri, '1' ler ise açık (on), doğru (true) gibi ifadeleri temsil eder.

3.10.1. BOOLE CEBRİ KURALLARI

Kural 1: 0 ve 1 ile yapılan işlemler

0 .0 = 0	0+0 = 0
0 .1 = 0	0+1 = 1
0.A = 0	0+A = A
1 .1 = 1	1+1 = 1
1.A = A	1+A = 1

Örnek: 0+(A+B.C+C.D+E.F)=1.(A+B.C+C.D+E.F)=A+B.C+C.D+E.F

Kural 2: Benzerlik Kuralı (Identity Law)

A+A = A	A.A = A
A+A+A+A+……+A = A	A.A.A.A……..….A =A

Örnek: $A.(A.B + C) = ?$

Örnek: $B.(A + B + C) = ?$

Örnek: $(A.A.\overline{B}.\overline{B} + C.C.C).(A.A.\overline{B}.\overline{B} + A.\overline{B}.\overline{B} + C.C) = ?$

Kural 3: **Tümleme Kanunu (Complementation Laws)**

$A + \overline{A} = 1$ $\qquad$ $A.\overline{A} = 0$

Kural 4: **Değişme Özelliği (Commutative Laws)**

A + B = B + A $\qquad$ A . B = B . A

Kural 5: **Birleşme Özelliği (Associative Laws)**

A+(B+C) = (A+B)+C = (A+C)+B = A+(C+B)

A.(B.C) = (A.B).C = (A.C).B = A.(C.B)

Kural 6: **Dağılma Özelliği (Distributive Laws)**

A.(B+C) = A.B + A.C

A+(B.C) = (A+B).(A+C)

Kural 7: **Yutma Özelliği (Absorption Law or Redundancy Law)**

A+A.B = A.(1+B) = A

A.(A+B) = A.A+A.B = A+A.B = A

Kural 8: **Çift Negatif Kuralı (Double Negative Law - Involution Law)**

$\overline{\overline{A}} = A$

Kural 9: **De Morgan Kuralı**

$\overline{A.B} = \bar{A} + \bar{B}$ $\qquad$ $\overline{A + B} = \bar{A}.\bar{B}$

$\overline{A.B.C} = \bar{A} + \bar{B} + \bar{C}$ $\qquad$ $\overline{A + B + C} = \bar{A}.\bar{B}.\bar{C}$

Örnek: $\overline{A.B + C.D + E.F} = ?$

Örnek: $\overline{(A + B).(C + D).(E + F)} = ?$

Kural 10:

$$A.B + A.\overline{B} = A \qquad (A + B).(A + \overline{B}) = A$$

Kural 11: Basitleştirme Kanunu (Minimisation Law)

$$A + \overline{A}.B = A + B \qquad A.(\overline{A} + B) = A.B$$

Kural 12: Konsensüs Teorisi (Consensus Theorem)

$$A.B + \overline{A}.C + B.C = A.B + \overline{A}.C \qquad (A + B).(\overline{A} + C).(B + C) = (A + B).(\overline{A} + C)$$

Tablo 1. Konsensüs Teorisinin Doğruluk Tablosu ile ispatlanması

A	B	C	AB	$\overline{A}$C	BC	AB + $\overline{A}$C + BC	AB + $\overline{A}$C
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	0	0	0	0
0	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	1	0	1	1	1

Kural 13: Transpozisyon Teorisi (Transposition Theorem)

$$A.B + \overline{A}.C = (A + C).(B + \overline{A}) \qquad (A + B).(\overline{A} + C) = (A.C) + (B.\overline{A})$$

Tablo 2. Transpozisyon Teorisinin Doğruluk Tablosu ile ispatlanması

A	B	C	AB	$\overline{A}$C	A+C	$\overline{A}$+B	AB + $\overline{A}$C	($\overline{A}$+B)(A+C)
0	0	0	0	0	0	1	0	0
0	0	1	0	1	1	1	1	1
0	1	0	0	0	0	1	0	0
0	1	1	0	1	1	1	1	1
1	0	0	0	0	1	0	0	0
1	0	1	0	0	1	0	0	0
1	1	0	1	0	1	1	1	1
1	1	1	1	0	1	1	1	1

Örnek: $\overline{A + \overline{B} + C \cdot \overline{D}}$ ifadesini Boole Cebri kurallarını kullanarak sadeleştiriniz?

Örnek: $F = A.\bar{B}.C + \bar{A}.B.C + B.\bar{C} + A.B.C$ ifadesini Boole Cebri kurallarını kullanarak sadeleştiriniz?

Örnek: $F = \bar{A}.B + \bar{A}.\bar{B} + A.B$ ifadesini Boole Cebri kurallarını kullanarak sadeleştiriniz?

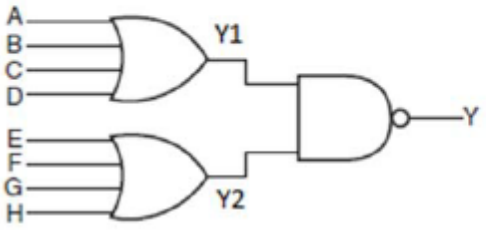
3.11. KOMBİNASYONLU (KARMAŞIK) LOJİK DEVRELER

Birden fazla lojik kapının kombinasyonlu (Karmaşık) olarak birbirine bağlanmasından oluşan devrelere Kombinasyonlu Lojik Devreler adı verilir. Bu devreler oluşturulurken veya devrenin çıkış fonksiyon değeri aranırken aşağıdaki sıralama izlenir.

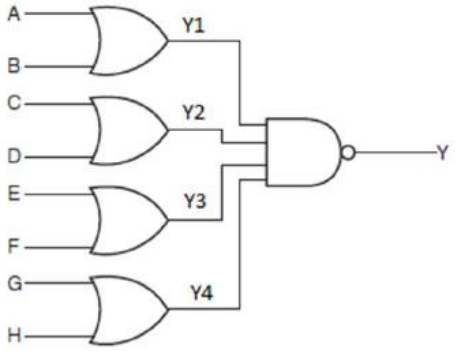
i) Lojik kapılı devre verilmişse verilen lojik devreden yararlanarak lojiksel ifadenin çıkarılması,
ii) Verilen, anlatılan veya var olan olayın lojik ifadesinin çıkarılması
iii) Çıkarılan lojik ifadenin sadeleştirilmesi,
iv) Sadeleştirilmiş ifadenin lojik kapılarla gerçekleştirilmesi, şeklinde bir senteze tabi tutulur.

3.11.1. Lojik Kapı Devresi Verilen Sistemin Lojiksel İfadesinin Bulunması

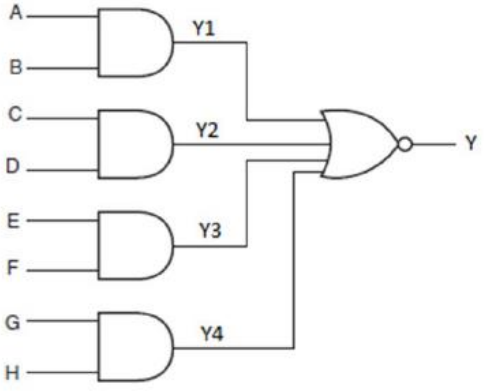
Örnek: Lojik kapı devresi verilen sistemin lojik ifadesini bulunuz?



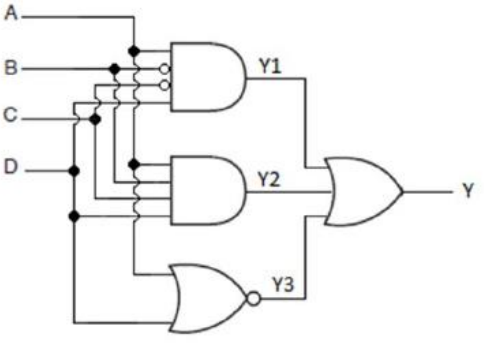
Örnek: Lojik kapı devresi verilen sistemin lojik ifadesini bulunuz?



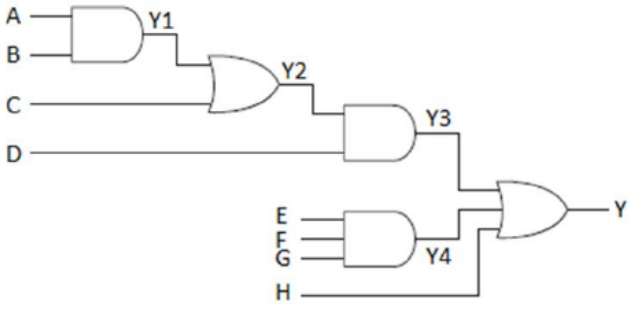
Örnek: Lojik kapı devresi verilen sistemin lojik ifadesini bulunuz?



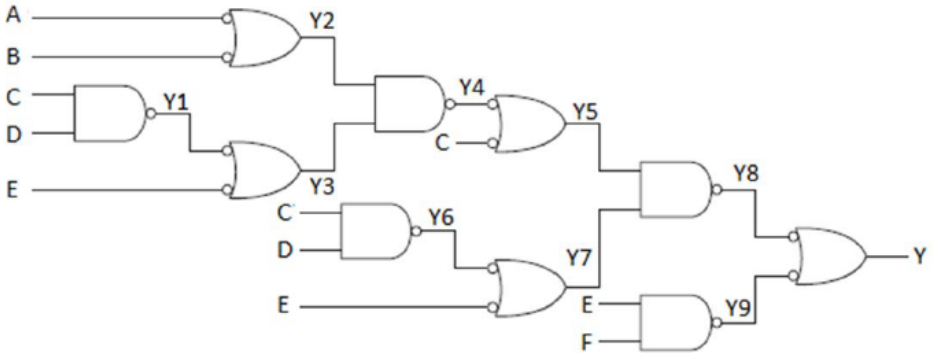
Örnek: Lojik kapı devresi verilen sistemin lojik ifadesini bulunuz?



Örnek: Lojik kapı devresi verilen sistemin lojik ifadesini bulunuz?



Örnek: Lojik kapı devresi verilen sistemin lojik ifadesini bulunuz?



3.11.2. Lojik Kapı Devresi Verilen Sistemin Lojiksel İfadesinin Bulunması

Herhangi bir olaydan veya verilen bir bağıntıdan lojik ifade çıkarılabilir. Olayda veya bağıntıda sonucu doğrudan etkileyen her faktör fonksiyonun birer lojik giriş değişkeni (ikili değişken)olur.

Örnek: Üç anahtar (S1, S2, S3) ile iki motorun (M1, M2) kontrolü yapılacaktır. Kapalı durumdaki anahtar sayısı tek olduğu durumda M1 motoru çalışacak, çift olduğu durumda M2motoru çalışacaktır. M1 ve M2 motorlarının lojik ifadesini bulunuz ve çiziniz?
(Anahtar Kapalı ≡ Lojik 1 , Anahtar Açık ≡ Lojik 0)

S1	S2	S3	M1	M2

M1 = ?

M2 = ?

Örnek: Üç bitlik iki tabanlı sayıların karesini veren sistemi lojik kapılar ile tasarlayınız?

GİRİŞ [X]				ÇIKIŞ [X^2]						
X	B2	B1	B0	A5	A4	A3	A2	A1	A0	X^2

Örnek: On tabanlı sayı ailesine ait rakamları Artı-3 (Excess-3) koduna çeviren sistemi lojik kapılar ile tasarlayınız?

GİRİŞ $[X]_{10}$					ÇIKIŞ $[X]_{+3}$				
X_{10}	B3	B2	B1	B0	A3	A2	A1	A0	X_{+3}

3.11.3. LOJİK İFADELERİN SADELEŞTİRİLMESİ

Elde edilen veya verilen lojik ifadelerin en kısa şekli en ideal durumdur. Çünkü ifadenin uzunluğu oranında bunu gerçekleştirecek lojik devredeki eleman sayısı artacaktır. Dolayısıyla devrenin maliyeti artacaktır. Bundan dolayı bazı işlem kuralları uygulanarak lojik ifadeler sadeleştirilir.

3.11.3.1. Cebirsel (Klasik) Sadeleştirme

Boole cebrinde anlatılan kurallar uygulanarak yapılan sadeleştirme şeklidir.

Örnek: $A.\bar{B} + B.\bar{C} + A.B.\bar{C} + \bar{A}.C$ ifadesini sadeleştiriniz?

$A.\bar{B}.(C + \bar{C}) + (A + \bar{A}).B.\bar{C} + A.B.\bar{C} + \bar{A}.(B + \bar{B}).C$

$A.\bar{B}.C + A.\bar{B}.\bar{C} + A.B.\bar{C} + \bar{A}.B.\bar{C} + A.B.\bar{C} + \bar{A}.B.C + \bar{A}.\bar{B}.C$

$A.\bar{B}.C + A.\bar{B}.\bar{C} + A.B.\bar{C} + \bar{A}.B.\bar{C} + \bar{A}.B.C + \bar{A}.\bar{B}.C$

$(A + \bar{A}).\bar{B}.C + A.(B + \bar{B}).\bar{C} + \bar{A}.B.(C + \bar{C}) = \bar{B}.C + A.\bar{C} + \bar{A}.B$

Örnek: $\bar{C}.D + B.\bar{C}.\bar{D} + A.\bar{B}.C + \bar{A}.\bar{B}.C.D + A.\bar{B}.D + A.\bar{B}.\bar{C}.\bar{D}$ ifadesini sadeleştiriniz?

$(\bar{A}.\bar{B} + A.\bar{B} + \bar{A}.B + A.B).\bar{C}.D + (A + \bar{A}).B.\bar{C}.\bar{D} + A.\bar{B}.C.(D + \bar{D}) + \bar{A}.\bar{B}.C.D +$

$A.\bar{B}.(C + \bar{C}).D + A.\bar{B}.\bar{C}.\bar{D} = \bar{A}.\bar{B}.\bar{C}.D + A.\bar{B}.\bar{C}.D + \bar{A}.B.\bar{C}.D + A.B.\bar{C}.D +$

$A.B.\bar{C}.\bar{D} + \bar{A}.B.\bar{C}.\bar{D} + A.\bar{B}.C.D + A.\bar{B}.C.\bar{D} + \bar{A}.\bar{B}.C.D + A.\bar{B}.C.D + A.\bar{B}.\bar{C}.D +$

$A.\bar{B}.\bar{C}.\bar{D} = \bar{B}.D + B.\bar{C} + A.\bar{B}$

Aşağıda, karşılaştırma yapmak için Zehirli Atık Yakma Fırınının Boole cebri ile sadeleştirilmesini tekrar ediyoruz.

$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$

↓ 1. ve 4. terimler BC parantezine alınır

$BC(\overline{A} + A) + A\overline{B}C + AB\overline{C}$

↓ $\mathbf{A + \overline{A} = 1}$ özdeşliği uygulanır

$BC(1) + A\overline{B}C + AB\overline{C}$

↓ 1A = A özdeşliği uygulanır

$BC + A\overline{B}C + AB\overline{C}$

↓ 1. ve 3. terimler B parantezine alınır

$B(C + A\overline{C}) + A\overline{B}C$

↓ $C + A\overline{C}$ ye $\mathbf{A + \overline{A}B = A + B}$ kuralı uygulanır

$B(C + A) + A\overline{B}C$

↓ Terimler dağıtılır

$BC + AB + A\overline{B}C$

↓ 2. ve 3. terimler A parantezine alınır

$BC + A(B + \overline{B}C)$

↓ $B + \overline{B}C$ ye $\mathbf{A + \overline{A}B = A + B}$ kuralı uygulanır

$BC + A(B + C)$

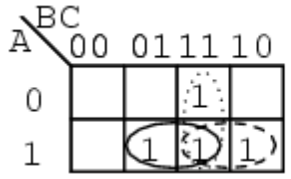
↓ Terimler dağıtılır

$BC + AB + AC$

yada Sadeleştirilmiş sonuç

$AB + BC + AC$

Aşağıda, yukarıdaki Boole cebri ile yapılan sadeleştirme ile karşılaştırma yapmak için Zehirli Atık Yakma Fırınının Karnaugh haritası çözümünü tekrar ediyoruz. Bu durum Karnaugh haritasının mantık sadeleştirme için neden çokça kullanıldığını gösteriyor.



Çıkış = AB + BC + AC

Karnaugh haritası metodu, önceki sayfadaki Boole cebrinden daha kolay gözükmektedir.

Lojiksel işlem kuralları uygulanarak yapılan sadeleştirme, hem zor ve hem de hesaplamasüresi uzun olduğundan dolayı genelde **KARNAUGH** haritasıyla sadeleştirme kullanılır.

3.11.3.2. Temel Açılımlar ve Standart İfadeler

Daha önceki konularda bahsedildiği üzere, bir binary değişkeni, ya kendi normal formu olan A olarak veya değili olan A' formu ile ifade edilebilir. Bu formlarla ifade edilebilendeğişkenler fonksiyon halini aldığı zaman; '**canonical form**' (kanun-kaide) olarakadlandırılan '**minterm'** (çarpımların toplamı) veya '**maxterm'** (toplamların çarpımı)modellerinden biri ile gösterilirler.

Değişken			Mintermler		Maxtermler	
A	B	C	Terim	İsim	Terim	İsim
0	0	0	A'B'C'	m_0	A+B+C	M_0
0	0	1	A'B'C	m_1	A+B+C'	M_1
0	1	0	A'BC'	m_2	A+B'+C	M_2
0	1	1	A'BC	m_3	A+B'+C'	M_3
1	0	0	AB'C'	m_4	A'+B+C	M_4
1	0	1	AB'C	m_5	A'+B+C'	M_5
1	1	0	ABC'	m_6	A'+B'+C	M_6
1	1	1	ABC	m_7	A'+B'+C'	M_7

Bir boolean ifadede bulunan değişkenlerin sahip olduğu veya oluşturabileceğikombinasyonların 'VE' (çarpım) işlemi sonucunda 1 olacak şekilde uyarlanmasına(değişkenin değeri 1 ise olduğu gibi alınıp, 0 ise değili ile ifade edilerek), '**minterm'** denir.Aynı yolla, değişkenlerin kombinasyonlarının 'VEYA' (toplama) işlemi sonucunda 0değerini almasını sağlayacak şekilde değişkenlerin şekillendirilmesine '**maxterm'** denir.

F (A, B, C, D)=$\sum_m(0,1,5,9) = m_0 + m_1 + m_5 + m_9$ → Çarpımların toplamı SOP (Sum of Products)

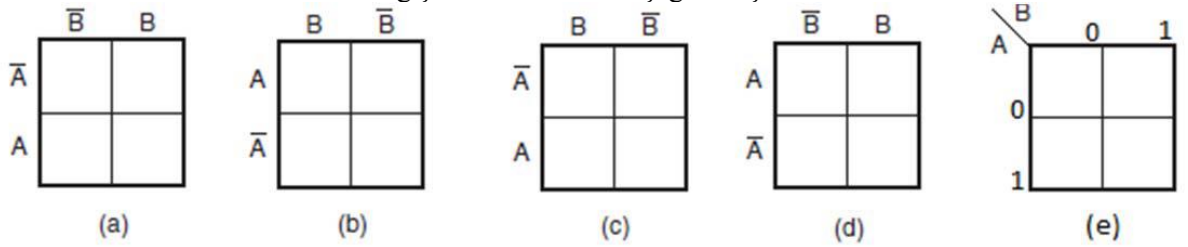
F (A, B, C, D)=$\prod_M(4,5,7,10) = M_4 . M_5 . M_7 . M_{10}$ → Toplamların çarpımı POS (Product of Sums)

3.11.3.3. Karnaugh Haritası ile Lojik İfadelerin Sadeleştirilmesi

Karnaugh haritası lojik ifadelerin grafiksel olarak gösterilmiş halidir. Başka bir ifade ile bazıözellikleri olan bir tablodur. Bu özellikler sayesinde lojik ifadeler kısa süre içerisinde en sadehale getirilir. Karnaugh haritasında sadeleştirilen ifade genelde en kısa şekildir. Başka yol veyöntemle daha fazla sadeleştirme yapılamaz. Max ve Min ifadeler doğrudan karnaughharitasına aktarılabilir. Ayrıca doğruluk tablosunda bulunan sonuçlar da kolayca karnaughharitasına aktarılıp sadeleştirilebilir.

İki değişkenli Karnaugh Haritası

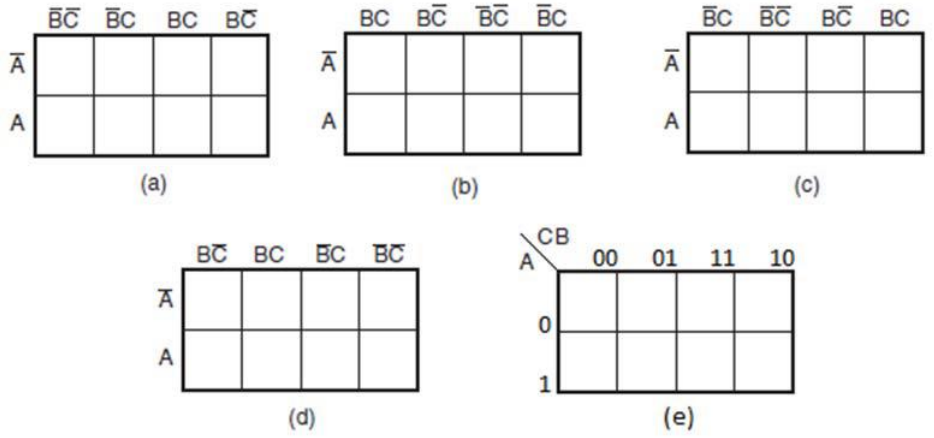
Karnough haritasını çizmek için kare şeklindeki tabloyu değişken sayısı ile orantılı olarakkarelere ayırmak gerekir. Değişken sayısı n ise kare sayısı 2^n formülünden bulunabilir.Dolayısıyla n=2 için 4 küçük kareye bölünmesi gerekir. Karnaugh haritasının kenarlarındakisatır ve sütunlara değişkenler cinsinden aşağıdaki şekilde isimler verilir.



Yukarıdaki şekilden de görüldüğü iki değişkenli karnaugh haritası 5 farklı şekildegösterilmektedir. Lojik devreler dersinde (e) seçeneğindeki gösterim kullanılacaktır.

Üç değişkenli Karnaugh Haritası

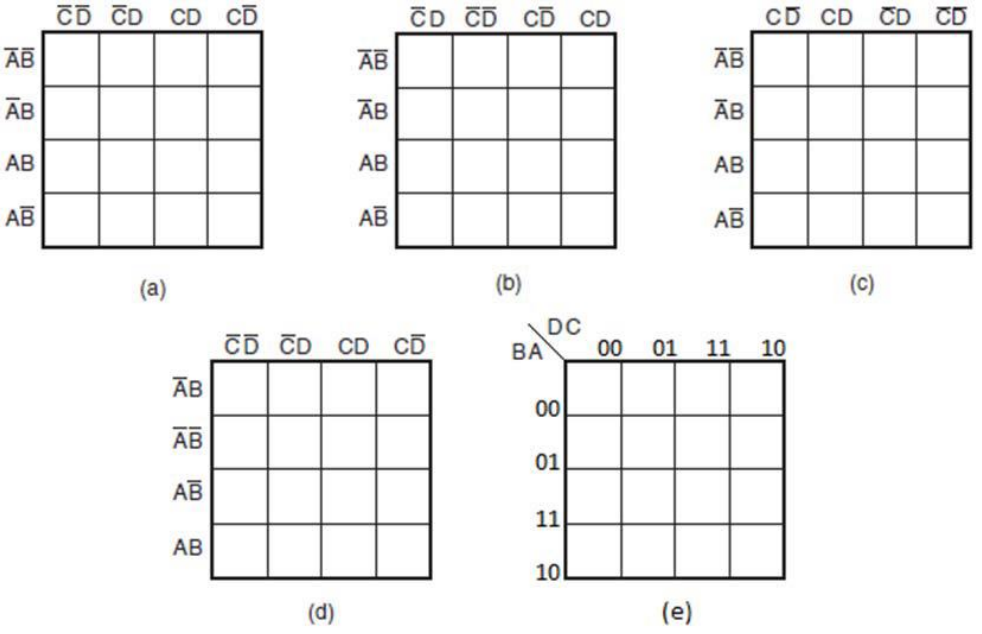
Değişken sayısı n=3 olduğu için tablonun 8 küçük kareye bölünmesi gerekir.



Yukarıdaki şekilden de görüldüğü üç değişkenli karnaugh haritası 5 farklı şekildegösterilmektedir. Lojik devreler dersinde (e) seçeneğindeki gösterim kullanılacaktır.

Dört değişkenli Karnaugh Haritası

Değişken sayısı n=4 olduğu için tablonun 16 küçük kareye bölünmesi gerekir.



Yukarıdaki şekilden de görüldüğü dört değişkenli karnaugh haritası 5 farklı şekildegösterilmektedir. Lojik devreler dersinde (e) seçeneğindeki gösterim kullanılacaktır.Karnaugh haritasındaki küçük karelerin değişkenler cinsinden neler ifade ettiği ve küçükkarelerin numaraları aşağıdaki şekilde verilmiştir.

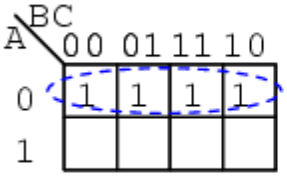
Dec Sıra	Binary (MSB) D	C	B	(LSB) A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

BA \ DC	00	01	11	10
00	0000	0100	1100	1000
01	0001	0101	1101	1001
11	0011	0111	1111	1011
10	0010	0110	1110	1010

BA \ DC	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

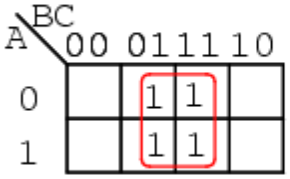
ÖRNEKLER

Çıkış = $\overline{A}B\overline{C} + \overline{A}B\overline{C} + \overline{A}BC + \overline{A}B\overline{C}$



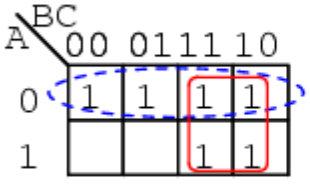
Çıkış = $\overline{A}$

Çıkış = $\overline{A}\overline{B}C + \overline{A}BC + A\overline{B}C + ABC$



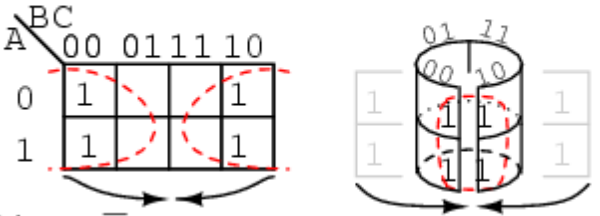
Çıkış = C

Çıkış = $\overline{A}B\overline{C} + \overline{A}B\overline{C} + \overline{A}BC + \overline{A}B\overline{C} + ABC + AB\overline{C}$



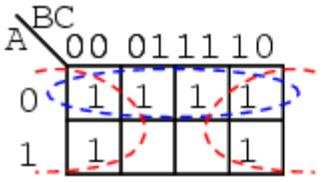
Çıkış = $\overline{A} + B$

Çıkış= $\overline{A}\,\overline{B}\,\overline{C} + A\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + A\,B\,\overline{C}$

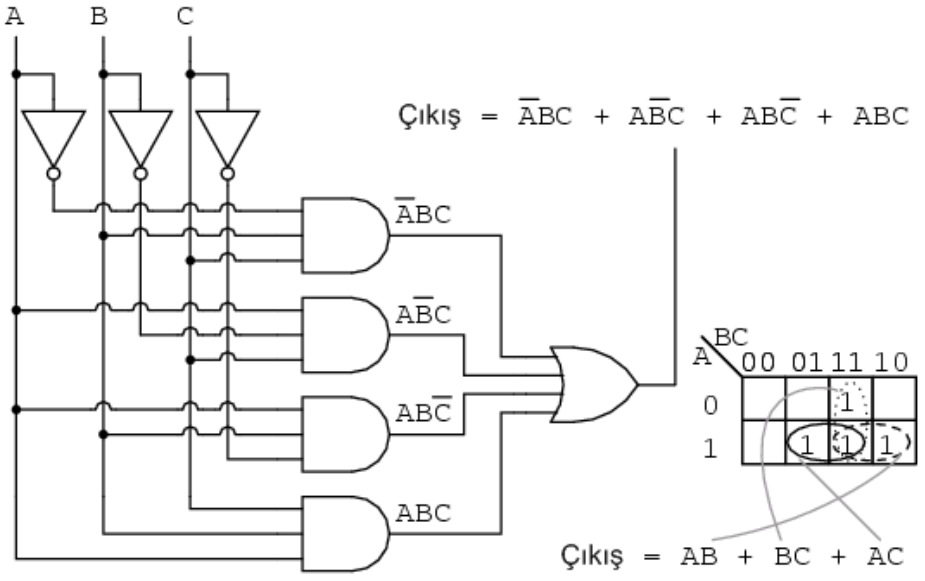


Çıkış= $\overline{C}$

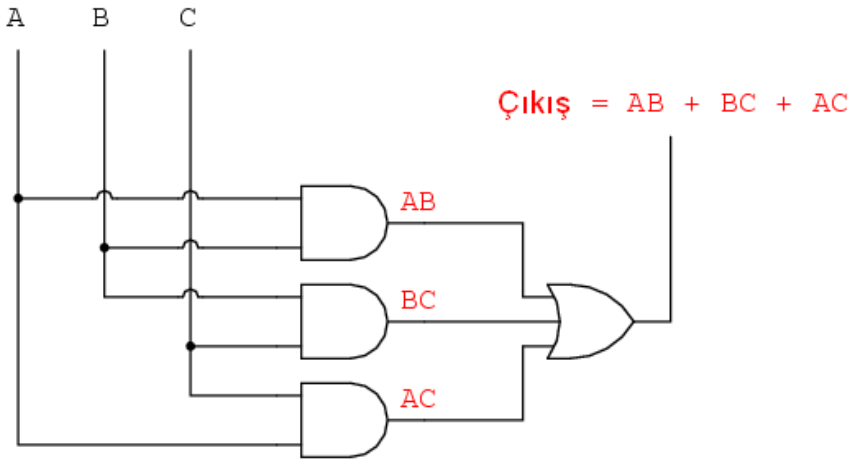
Çıkış= $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + A\,B\,\overline{C}$



Çıkış= $\overline{A} + \overline{C}$



Çıktı olan Boolean denklemi dört çarpım terimine sahiptir. Dört 1'i karşılık gelen çarpım terimleri ile eşleştirin. Hücre gruplarını oluşturarak, ikili üç grup elde ederiz. Her bir grup için bir adet çarpım terimi olmak üzere sadeleştirilmiş sonuçta toplam üç çarpım terimi olacaktır. Aşağıda gösterilen sonucun kapı diyagramı için, Boole cebri bölümünden "Zehirli Atık Yakma Fırını" konusuna bakınız.



ÖRNEKLER

Örnek: $F = A.B.\bar{C}.\bar{D} + \bar{A}.B.\bar{C}.\bar{D} + A.\bar{B}.C.D + A.\bar{B}.C.\bar{D} + \bar{A}.\bar{B}.C.D + A.\bar{B}.C.D$
Verilen lojik ifadeyi karnaugh haritasına yerleştiriniz?

Örnek: $F = \bar{C}.D + B.\bar{C}.\bar{D} + A.\bar{B}.C + \bar{A}.\bar{B}.C.D + A.\bar{B}.D + A.\bar{B}.\bar{C}.\bar{D}$
Verilen lojik ifadeyi karnaugh haritasına yerleştiriniz?

Örnek: $F\,(A, B, C, D) = \sum_m(0, 1, 5, 9, 11, 15)$ Verilen MIN ifadeyi karnaugh haritasına yerleştiriniz?

Karnaugh Haritası Farketmez (Don't Care) İfadesi

Lojik fonksiyonlarda ifadelerden bazıları fonksiyonun alacağı değere bağlı değildir. Yani buifadeler ne olursa olsun fonksiyonun sonucuna etki etmez. Böyle ifadelere farketmez (Don'tCare) denir. Karnaugh haritasında Don't Care ifadeler 'X' ile gösterilir. Bu ifadeler karnaughharitasındaki gruplamalarda 0 veya 1 olarak alınabilir. Karnaugh haritasındaki her bir Don'tCare ifade gruplamada

olmak zorunda değildir. Sabit 0 veya 1 için geçerli olan bir grubamutlaka ait olma zorunluluğu Don't Care için geçerli değildir.

Örnek: $F(A,B,C,D) = \sum_m(0,1,5,9,11,15) + d(2,3,7)$ Verilen ifadeyi karnaugh haritasına yerleştiriniz?

Karnaugh Haritasında Gruplama Yapılırken Dikkat Edilecek Hususlar

1- Üst ve alt kenarlar katlandığında karelerin içindeki doğruluk değerleri üst üste gelirsebirbirine komşu olmaktadır.

2- Sağ ve sol köşeler katlandığında karelerin içindeki doğruluk değerleri üst üste gelirsebirbirine komşu olmaktadır.

3- Köşeler katlandığında karelerin içindeki doğruluk değerleri üst üste gelirse dört köşekomşu olmaktadır.

4- Gruplama yaparken Lojik 0' lar yada Lojik 1' ler gruplandırılır.

5- Lojik 1' ler gruba alınırken de Karnaugh haritasındaki Lojik 1' lerin tamamı seçilmeli, açıktaLojik 1 kalmamalıdır. Lojik 1' ler gruplandığı zaman çıkan lojiksel ifade, **Çarpımların Toplamı**(SOP) şeklinde olmaktadır.

6- Lojik 0' lar gruba alınırken Karnaugh haritasındaki Lojik 0' ların tamamı seçilmeli, açıktaLojik 0 kalmamalıdır. Lojik 0' lar gruplandığı zaman çıkan lojiksel ifade, **Toplamların Çarpımı**(POS) şeklinde olmaktadır.

7- Don't Care ifadeler 'X', karnaugh haritasındaki gruplamalarda 0 veya 1 olarak alınabilir.Karnaugh haritasındaki her bir Don't Care ifade gruplamada olmak zorunda değildir.

8- Gruplama yaparken 2^n' li gruplar (1, 2, 4, 8, 16, vs. gruplar) oluşturulmalıdır. Bu rakamlarındışında yapılan ve ifade edilen gruplar yanlış gruplama olur.

9- Gruplamada ilk önce 2^n değerinin en büyük değerine sahip gruplar oluşturulur.Oluşturulamıyorsa bir sonraki küçük 2^n'li gruplar oluşturulur.

10- Karnaugh haritasının tamamı bir grup oluyorsa fonksiyon sıfır veya bir'dir.

11- Karnaugh haritasındaki sabit bir veya sıfır birden fazla grupta yer alabilir.

ÖRNEKLER

Örnek: $F = A.\bar{B}.\bar{C} + A.C + \bar{A}.B.\bar{C} + A.B$ Verilen lojik ifadeyi karnaugh haritası ile sadeleştiriniz?

Örnek: $F(A,B,C,D) = \sum_m(1,2,5,6,9) + d(10,11,12,13,14,15)$ Verilen ifadeyi karnaugh haritası ile sadeleştiriniz?

Örnek: $F(A,B,C,D) = \sum_m(0,2,3,5,6,7,8,10,11,14,15)$ Verilen ifadeyi karnaugh haritası ile sadeleştiriniz?

Örnek: $F(A,B,C,D) = \sum_m(2,4,6,8,9,10,12,13,14)$ Verilen ifadeyi karnaugh haritası kullanarak;

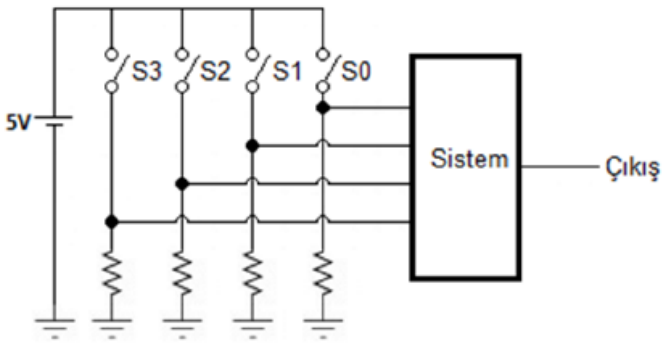
Örnek: $F(A,B,C,D) = \sum_m(8,9,10,11,12,13,14,15)$ Verilen ifadeyi karnaugh haritası ile sadeleştiriniz?

Örnek: $F(A,B,C,D) = \prod_M(0,1,2,3,4,5,6,7)$ Verilen ifadeyi karnaugh haritası ile sadeleştiriniz?

Örnek: $F = ACD + AB\overline{D} + BC\overline{D} + A\overline{C}\,\overline{D}$ Verilen ifadeyi karnaugh haritası ile sadeleştiriniz?

Örnek $F = \overline{A}C\overline{D} + A\overline{D} + B\overline{C}\,\overline{D} + AB\overline{C} + A\overline{C}D + A\overline{B}\,\overline{C}\,\overline{D}$ Verilen ifadeyi karnaugh haritası ile sadeleştiriniz?

Örnek:



Yandaki şekilde görülen devre bir fotokopi makinesinin kontrol devresidir. Makinenin içerisinde kağıdın yolu üzerinde 4 adet anahtar bulunmaktadır. Sistem herhangi iki anahtarın kapatılması durumunda çıkış vermektedir. Sisteme ait devreyi Karnaugh haritası kullanarak NOR kapılarıyla tasarlayınız?

Uygulamalı Örnek : Üç kapılı bir binada iki ve daha fazla kapının aynı anda açık olması istenmiyor. Bu durumun gerçekleşmesi durumunda ışıklı ve sesli ikaz veren devreyi kurunuz. Lojik devre tasarlarken aşağıdaki işlemler sırasıyla gerçekleştirilir.

1) Yapılacak devre için bütün detaylar gözden geçirilir.

2) Yapılacak işlemin doğruluk tablosu hazırlanır.

3) Doğruluk tablosundan çıkarılacak sonuçlar yazılır.

4) Çıkarılan sonuçlar karno haritası kullanılarak basitleştirilir.

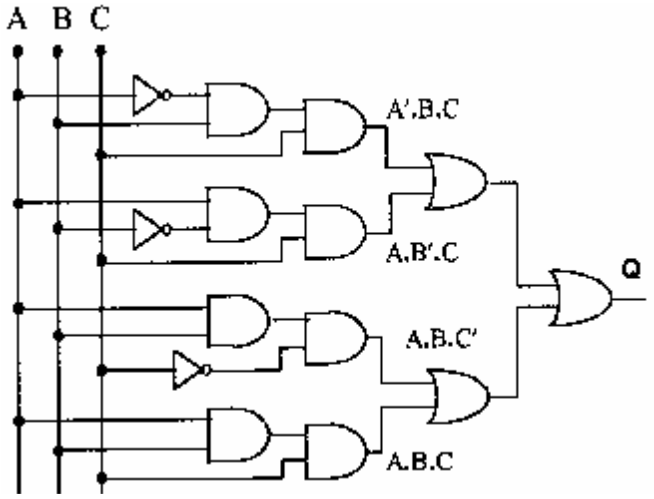
5) Lojik devre çizilir.

Yapılacak Devre Için Bütün Detayların Gözden Geçirilmesi

Tasarlanacak devrede kapı ve pencereler için anahtar kullanalım. Bu anahtarlar kapıve pencere ile birlikte açılıp kapanan anahtarlar olsun. Yani kapı ya da pencere açıksa,anahtar da açık olsun. Eğer kapı ya da pencere kapalı ise, anahtar da kapalı olsun. Devredekianahtarları A, B ve C değişkenleri gibi düşünelim.
Devrede çıkışa sesli ikaz devresi veya ışıklı ikaz düşünelim. Eğer ikaz durumugerçekleşirse (kapı ya da pencerelerden ikisi birlikte açık olursa) ikaz çalışsın yani çıkış 1olsun.

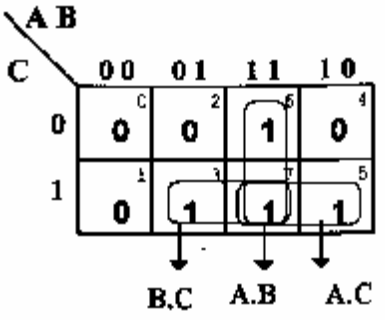
Q = (A'.B.C) + (A.B'.C) + (A.B.C') + (A.B.C)

Konum	Girişler			Çıkış	Girişler			Çıkış	
	A	B	C	Q	1. Pencere	2. Pencere	Kapı	İkaz	
0	0	0	0		Kapalı	Kapalı	Kapalı	Yok	
1	0	0	1		Kapalı	Kapalı	Açık	Yok	
2	0	1	0		Kapalı	Açık	Kapalı	Yok	
3	0	1	1	1	Kapalı	Açık	Açık	Var	Q=A'.B.C
4	1	0	0		Açık	Kapalı	Kapalı	Yok	
5	1	0	1	1	Açık	Kapalı	Açık	Var	Q=A.B'.C
6	1	1	0	1	Açık	Açık	Kapalı	Var	Q=A.B.C'
7	1	1	1	1	Açık	Açık	Açık	Var	Q=A.B.C

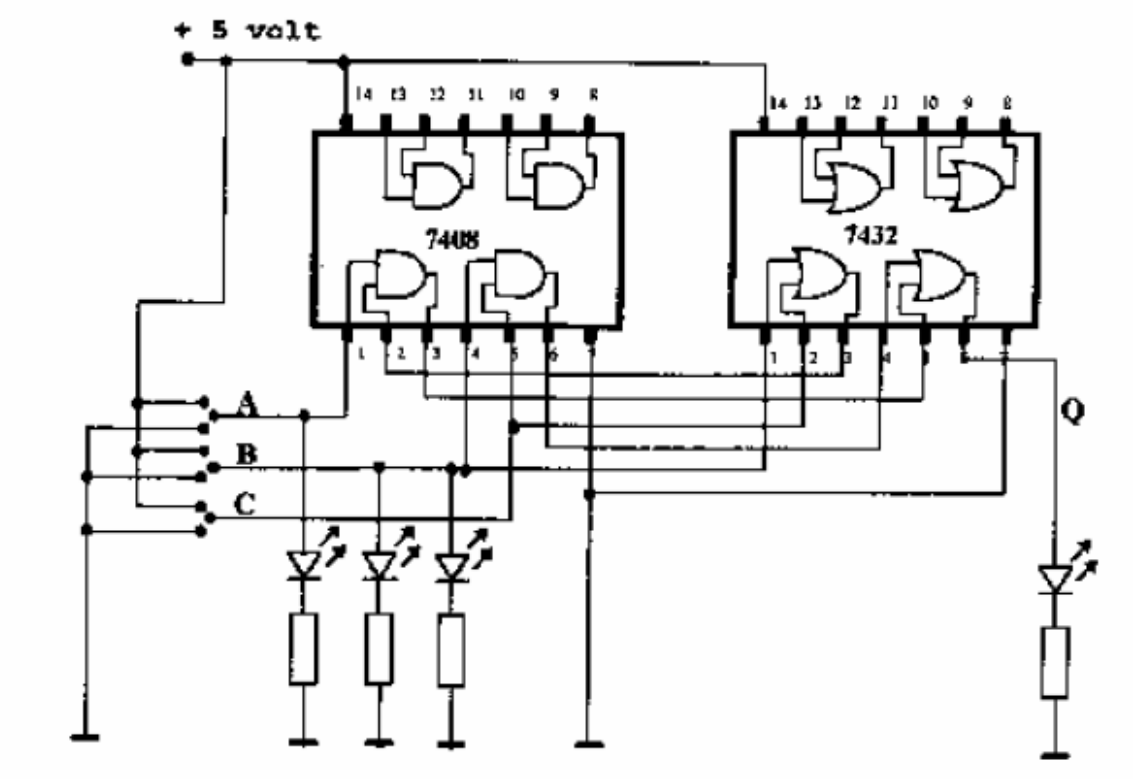


Çıkan Sonuçların Sadeleştirilmesi (Karno Haritası ile)

Kutu no:	GİRİŞLER			ÇIKIŞ
	A	B	C	Q
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



Sadeleştirmeyi yaptıktan sonra Q = (A.B) + (B.C) + (A.C) sonucu elde edilir. Busonucu da tekrar parantez kullanarak Q = (A.(B+C)) + (A.B) haline getirerek devreyiyeniden çizersek aşagıdaki gibi olur.
Sadeleştirilmiş devrede görüldügü gibi devrede kullanılan kapı sayılan 15'ten 4'edüşmüştür. Dolayısıyla kullanacağımız entegre de azalacaktır. Bir adet 7408 entegre ve biradet 7432 entegresi kullanılarak bu ikaz devresi kurulabilir.



➢ **Devre Şeması ve Malzeme Listesi**

1 adet bread board
4 Adet 390 ohm direnç
3 adet kırmızı led 1 adet yeşil led
5 volt DC güç kaynağı
1adet 7408 entegre
1adet 7432 entegre

4. Mikro Bilgisayar Sistemleri ve Assembler

4.1.Mikro Denetleyici Nedir?

- Tek bir silikon kılıf üzerinde toplanmış entegre devredir.
- Her yıl yüz milyonlarca adet mikrodenetleyici endüstri tarafından tüketilir.
- Alarmlı saatlerde, mikrodalga fırınlarda, bulaşık makinelerinde, buzdolaplarında v.b. bir cihazda kullanılmaktadırlar.
- Tek-çip bilgisayar, mikrobilgisayar veya yerleşik bilgisayar sistemleri isimleri altında da tanıtılmaktadır.
- Tek başlarına çalışabilirler
- Tek-çip devre elemanıdırlar
- Sistem kararları genellikle harici sinyallere bağlıdır
- Elektronik bir cihazın davranışlarını denetlerler ve kontrol ederler
- Bir devrenin beyni konumundadırlar

4.2.Mikro Denetleyiciyi Meydana Getiren Birimler

- Bir mikroişlemci çekirdeği (CPU)
- Program ve veri belleği (ROM, RAM)
- Giriş/Çıkış (I/O) birimleri
- Saat darbesi üreteçleri

- Zamanlayıcı/Sayıcı birimleri
- Kesme kontrol birimi
- A/D–D/A (Analog/Dijital–Dijital/Analog) çeviriciler
- Darbe genişlik üreteci (PWM)
- Seri Haberleşme Birimi (UART, RS-232, CAN, I2C vb.)
- Diğer çevresel birimler.

Mikrodenetleyici temel olarak dört bileşenden oluşur

1. Mikroişlemci
2. Bellek
3. Giriş/çıkış birimi
4. Saat darbe üretici

4.3.Mikro İşlemciler

Mikroişlemciler modern bilgisayar sistemlerinin beyni olarak kabul edilmektedir. Bilgisayarlar için son derece önemli olan bu sistemler 1971 yılı itibarı ile Intel firmasının 4004 ve Texas Instruments firmasının TMC1795 işlemcilerinin üretilmesi başlamıştır. Motorola firması ise 1974'ün sonlarında 6800 ile ilk mikroişlemci modelini üretmiştir.

İntelin ilk işlemcisi 4004

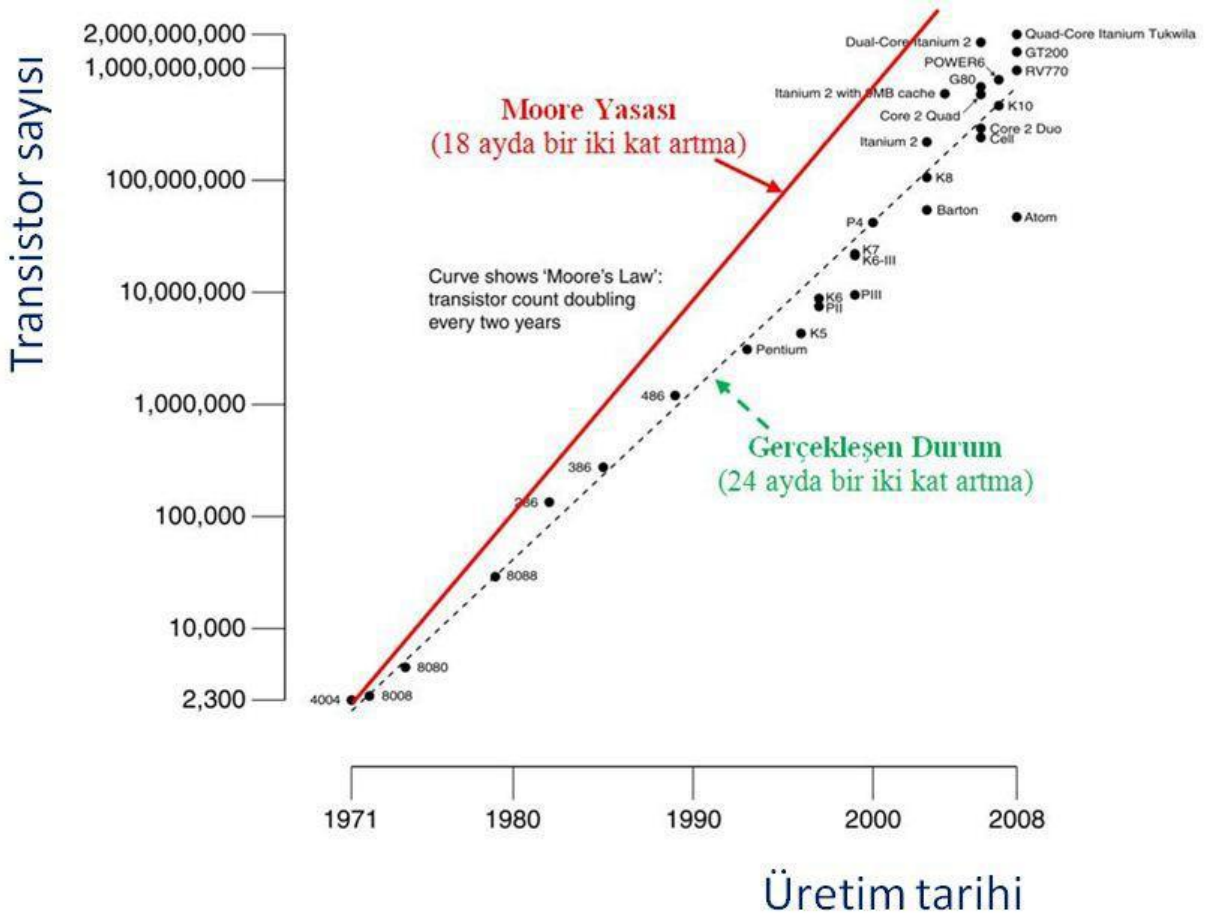
Intel firmasının halen en büyük rakipleri arasında yeralan AMD (Advanced Micro Devices) firması ilk mikroişlemci ürünü olan 29000'u 1988 yılında üreterek mikroişlemci pazarına girmiştir.

1990'lı yılların ilk yarısına gelindiğinde, Intel firması birçok mikroişlemci ile sektöre hizmet etmekteydi. 386SL, 486SL, 960SA, 960CA, 860XP, 960CF, Pentium, DX4, 960Jx, Pentium Pro gibi mikroişlemcileri üretmiştir. 1990'lı yılların başında IBM firması 386SLC, RSC, 486SLC, Power 2, 486SX2, P2SC gibi mikroişlemcileri piyasaya sürmüştür. Bu dönemde faaliyete geçen bir diğer önemli mikroişlemci üreticisi ise PowerPC (**P**erformance **o**ptimization **w**ith **e**nhanced **R**ISC – **P**erformance **C**omputing) dir. Apple, IBM ve Motorola firmaları ortaklaşa olarak PowerPC birliğini kurmuştur.

4.4.MOORE YASASI (MOORE'S LAW)

Mikroişlemci dünyası açısından önemli bir yere sahip olan Intel firmasının kurucuları arasında olan Dr. Gordon MOORE mikroişlemciler hakkında yaptığı çalışmalar üzerine 1965 yılı Nisan ayında yayınlamış olduğu bir makale ile bilgisayar tarihine kendi soyismi ile anılan bir yasa üretmiştir. Dr.Moore yaptığı öngörüsüne göre her 18 ayda bir, transistor sayısının iki kat artacağını düşünmüştür. Yıllar içinde hayata geçen mikroişlemci yapı ve özellikleri incelendiğinde, üretilen mikroişlemci özelliklerinin Moore yasası ile aynı oranla olmasa da, bu yasaya benzer bir artış şekline sahip olduğu gözlemlenmiştir.

Aşağıda x86 tabanlı mikroişlemciler ile Moore yasasının öngörü kıyaslanması görülmektedir.



x86 tabanlı mikroişlemcilerin yıllar içerisinde her 24 ayda iki kat arttığı görülmektedir. Moore yasasında 18 ayda bir gözüken artış aynı olmamakla birlikte benzer bir eğilim göstermiştir.

Mikro bilgisayar sistemlerinde en temel kullanılan mimariler RISC ve CISC dir.

4.5.RISC MİMARİSİNİN GENEL ÖZELLİKLERİ

RISC mimarisi; aralarında *Sun SPARC, PowerPC, MIPS, DEC Alpha* gibi bilgisayar sistemlerinin seçmiş oldukları bilgisayar mimari yöntemidir. Bu yöntem İlk olarak Motorolafirmasının 1974'ün sonlarında ürettiği 6800 mikroişlemcisinde kullanılmıştır.
RISC kelimesi bir kısaltmadan gelmekte olup, **R**educed **I**nstruction **S**et **C**omputer kelimelerinin baş harflerinden oluşmaktadır. İngilizce RISC kısaltmasının Türkçe karşılığı "**A**zaltılmış **K**omut **K**ümesi **B**ilgisayarı" şeklindedir. Burada komut kümesinin azaltılmasından kastedilen olay sistemin performansının yükseltilmesi için basit ve az sayıda komut listesinin (kümesi) bulunmasıdır.

Diğer taraftan, RISC mimarisi kullanan bilgisayar sistemlerinin birçoğunun komut kümesi oldukça fazladır. Bu durum bazı durumlarda kafa karışıklıklarına sebep olmaktadır.
Az sayıda bulunan komut listesi sayesinde komutların uzunluğu ve yapısı genellikle aynıdır. Bu şekilde mikroişlemcilerin çalışma prensiplerinde önemli bir yer tutan "pipelining (grup halinde komut işleme hattı)" uygulamasını gerçekleştirmek az sayıda bulunan komut sayesinde kolaylaşmaktadır.

RISC mimarisinde yazılan programlarda bir işlemi yapmak için birkaç işi bir arada yapan bir komut yerine (kompleks bir işlem yerine), basit işlemler içeren birkaç komut arka arkaya kullanılmaktadır. Bu durumda yazılan Assembly programının satır uzunluğu daha fazla olmaktadır.
RISC mimarisinin başarısını arttırmak için "pipelining" ve "superscalar" uygulamalarının kullanılması gereklidir.
RISC mimarisinde komutların şekli basit ve aynı tarzda olduğundan, komutların kodunun çözülmesi (decode) kolay bir şekilde gerçekleştirilmektedir.
RISC mimarisinin özellikleri arasında yapısının kompleks olmaması (basit) olması bulunmaktadır. Bu sayede, speedup (ölçeklenebilirlik) özelliği açısından avantaj kazanılmaktadır. Kompleks olmaması sayesinde sistem üzerinde yaşanacak hata olasılığı da azaltılmıştır.
Ayrıca, RISC mimarisinin basitliği sayesinde, sistemde bulunan "register (yazmaç)" ve "cache (önbellek)" sayısı daha fazla olabilmektedir. Diğer bir değişle, sistemde daha fazla register ve cache için yer kalabilmektedir.
RISC mimarisinde yazılımdan çok şey beklendiği için derleyiciler (compiler) daha karmaşık yapıdadır.
RISC mimarisi hakkında daha fazla bilgi MIPS Assembly programlama ve CISC ve RISC mimarilerinin karşılaştırmalarının verileceği haftalarda gösterilecektir.

4.5.1. GÜNÜMÜZDE RISC MİMARİSİNİ KULLANAN SİSTEMLER

Yakın bir zamana kadar bilgisayar sistemlerinde gerek masaüstü, gerek Laptop, gerekse de sunucu gibi sistemlerde RISC mimarisi yoğun olarak bulunmaktayken, şu anda daha bu mimarinin kullanımı bu alanlarda önemli ölçüde azalmış, yukarıda örnekleri şekillerle de gösterilen oyun konsolları, cep telefonları ve otomobillerde bulunan işlemcilerde önemli ölçüde sürdürmektedir.

Örneğin, Macintosh bilgisayar sistemleri yakın bir tarihe kadar RISC mimarisini kullanan PowerPC işlemcilerini kullanmışlardır. 2006 yılı ile birlikte Macintosh bilgisayarlarda Intel üretimi olan x86 tabanlı işlemci sistemleri kullanılmaya başlanmıştır.

Xbox 360 (IBM PowerPC tabanlı, 3 çekirdekli, Xenon, 3.2 GHz) Xbox 360 konsolu için tasarlanmış olan Xenon işlemci kullanılmaktadır

Playstation 3 (Cell Broadband Engine, 3.2 GHz), Sony Playstation 3 konsolu, Cell Broadband Engine ismindeki Sony, Toshiba ve IBM tarafından hazırlanan 3.2 GHz hızındaki işlemciyi kullanmaktadır.

Nintendo Gamecube (IBM PowerPC, Gekko, 486 MHz), Nintendo Gamecube konsolu RISC mimarisi kullanan sistemlerden birisidir. İçinde IBM PowerPC, Gekko, 486 MHz işlemci kullanılmaktadır

Nokia 5800 XpressMusic, ARM 11 CPU, 369 MHz, **Nokia 3230, 32bit ARM9 CPU, 123 MHz**
Nokia 5800 XpressMusic, ARM 11 CPU, 369 MHz işlemci kullanmaktadır. Nokia 3230, 32bit ARM9 CPU, 123 MHz işlemci kullanmaktadır

Otomobillerde bulunan mikroişlemci sistemlerinde RISC mimarisi kullanılmaktadır.

SORULAR

1. İngilizce "*RISC*" kısaltmasının açılımı nedir?
a. Reduced Instruction Set Computer
b. Risk Indicator Security Computer
c. Recycle Instructions Science Computer
d. Risk Indicator Science Complex
e. Recycle Information Simulator Computer

2. İngilizce "*RISC*" kısaltmasının Türkçe açılımı nedir?
a. Resesyon Belirteci Sanılan Bilgi
b. Azaltılmış Komut Kümesi Bilgisayarı
c. Azaltılacak Belirteç Kümelenir Bilgisi
d. Radikal İkilik Sistem Bilgisayarı
e. Okuma Öncelikli Bilimsel Bilgisayar

3. Günümüzdeki durum değerlendirildiğinde, aşağıdaki bilgisayar sistemlerinden hangisinde RISC mimarisi kullanılma yoğunluğu en azdır?
a. Giyilebilir bilgisayar sistemleri
b. Oyun konsolu
c. Masaüstü bilgisayar
d. Cep telefonu
e. Otomobil içi gömülü bilgisayar sistemleri

****

4.6.MIPS ASSEMBLY PROGRAMLAMA

Bu hafta mikrobilgisayar sistemlerinin yapısal iki temel grubundan birisi olan "***RISC (Reduced Instruction Set Computer)***" mimarisi üzerinde kullanılan MIPS Assembly programlama dilinin genel özellikleri hakkında bilgi verilecektir.

4.7.ASSEMBLY DİLİNİN ORTAYA ÇIKIŞ NEDENİ

İnsanlar kendi aralarında iletişim kurmak için "*Türkçe*", "*İngilizce*", "*Fransızca*", "*Almanca*" ve "*Yunanca*" gibi insani dilleri kullanmaktadır. Diğer taraftan dijital bilgisayar sistemleri ise "0" ve "1" mantığı üzerine çalışmaktadır. Günümüzde yazı, yazmaktan, oyun oynamaya, simülasyon programlarından, sağlık uygulamalarına kadar bütün dijital bilgisayar sistemleri 0 ve 1 mantığı kullanmaktadır. İkilik (binary) sistem şeklinde anılan bu sistemler bilgisayarların temel yapısını oluşturmaktadır.

İşte bahsedilmiş olan bu ikilik sistem mantığı ile çalışan dijital bilgisayarlar, en alt seviyede incelendiğinde yalnızca "0" ve "1" den anlamaktadır. Bilgisayar Biliminde 0 ve 1'li sistem kullanılan dil makine dili olarak bilinmektedir. Dolayısı ile dijital bilgisayarlar üzerinde yapılmak istenen işlemlerin direkt olarak makine dilindeki komutlardan oluşması gerekmektedir.
kısa bir makine dili programı yazılmak istendiğinde bile, bir sürü ikilik sistemde sayı ile uğraşmak gerekmektedir .İşte bu örnekte de gösterilen, insanlar açısından sevimsiz olan ikilik sistemdeki kodlarla uğraşmak yerine insanlara daha uygun bir yöntem bulunması hedeflenmiştir.
Buradan hareketle, insanların kullanmakta olduğu iletişim yöntemine daha yatkın bir yöntemin kullanılması gerek programların yazılmasında, gerek programlama sırasında ortaya çıkan problemlerin düzeltilmesinde, gerekse de başkaları tarafından yazılmış olan programların anlaşılması açısından büyük yarar sağlayacağı görülmüştür.

Bilgisayar Bilimindeki gelişmelerin büyük çoğunluğu İngilizce dili kullanılarak gerçekleştirilmiştir. Ana dili İngilizce olmayan kişiler bile genellikle çalışmalarını İngilizce dilinde yürütmektedir. Bu gerçekten ötürü İngilizce dili bir bakıma Bilgisayar Biliminde baskın bir rol oynamıştır.

İngilizcenin baskın olma gerçeği Assembly dili için de geçerli olmuştur. Örneğin MIPS Assembly dili İngilizce kelimelerden ve de İngilizce'de bulunan kelimelerin çeşitli kısaltmalarından oluşmaktadır.

MIPS Assembly dilindeki üç komuttan birincisi olan "add" toplama işleminde kullanılmaktadır. Bilindiği gibi İngilizce dilinde "add" kelimesi ilave etmek anlamına gelmekte, diğer bir değişle toplama anlamını da ifade etmektedir.

İkinci komut olan "sw" ise yazmaçtaki (register) bilgileri hafızaya yükleme işlemi yapmaktadır. Burada gösterilen "sw" harfleri "**s**tore **w**ord" kelimelerinin baş harflerinden oluşmaktadır.

Üçüncü komut olan "lw" ise hafızadan bir bilgiyi yazmaça (register) yüklemektedir. Bu bağlamda "lw" harfleri "**l**oad **w**ord" kelimelerinin baş harflerinden oluşmaktadır.

Dolayısı ile MIPS Assembly dilinde hazırlanmış olan bir program ikilik sistemde artarda gelen ve ilk bakıldığında insanlar açısından pek bir anlam ifade etmeyen sayılar yerine, insanların kullanmakta olduğu bir dil olan İngilizce merkezli bir yapıda yer alarak adaptasyonu kolaylaştırmaktadır.

Etiket komut yorum

MIPS Assembly dilinde bir satırın genel yapısı

Etiket olarak gösterilen kısım (kırmızı renkte gösterilen) yazılan MIPS Assembly programında belirli bir noktayı göstermektedir. Programın çalışması sırasında farklı noktalara direkt olarak erişmek için kullanılmaktadır. Bir etiket (label) bir programda yalnızca bir yerde kullanılabilir. Örneğin on ikinci satırda "BASLAT" isimli bir etiket kullanılması durumunda, programın başka hiçbir yerinde "BASLAT" ismi ile etiket kullanmak mümkün değildir.

Etiketler büyük harf/küçük harf farkına duyarlı yapıdadır. Buradan hareketle, bir satırda "BASLAT" isimli bir etiketi olursa, başka bir satırda "Baslat" isimli bir etiket bulunabileceği gibi, diğer bir satırda ise "BaSLaT" isimli etiket bulunabilmektedir.

MIPS Assembly dilindeki bir satırın ikinci ve en önemli olarak kabul edilecek kısmı satırın komutudur (yeşil renkle gösterilen yer). Bir satırın komut kısmında gösterilen işlem, bilgisayar tarafından o aşamada yapılması istenen adımdır.

Bir MIPS Assembly satırının komut kısmında "*add*", "*sub*", "*mult*", "*div*", "*lw*", "*sw*", "*beq*" ve "*jr*" komutlarının da aralarında bulunduğu çeşitli komutlardan birisi kullanılmaktadır.

Farklı türlerde komutlar bulunmaktadır. O satırda kullanılan komutun yapısına göre "*yazmaçlar (registers)*", "*tamsayı (integer)*", "*onaltılık sistemde sayı (hexadecimal number)*" ya da "*bir etiket*" işleme yön veren elemanlar olarak bulunabilir.

Örneğin, toplama işlemi yapan "add" komutu üç yazmacı (register) işleme yön veren değer olarak almaktadır. Diğer taraftan "addi" komutu ise iki yazmaç (register) ve bir tamsayıyı işleme yön veren değer olarak almaktadır. Diğer taraftan "j" komutu ise işlemini yürütmek için bir etiket kullanmaktadır.

MIPS Assembly dilindeki bir satırın genel yapısındaki üçüncü ve son bölüm ise satırda yer alabilecek "yorum" kısmıdır (koyu mavi ile gösterilen). Yorum kısmı tamamen programcının isteğine bağlı olarak yazabileceği bir kısımdır. Bu kısmın amacı programın okunması ve anlaşılmasını kolaylaştırıcı "yorumların (comment)" yazılmasına olarak verilmesidir.

4.8.MIPS Assembly Dilinde Örnek Programlar

Makine dilindeki 0 ve 1'lerden oluşan programlama yapısı insanlara soğuk gelmektedir. Bu soğukluğu bir ölçüde bertaraf edebilmek için Assembly dili kullanılmaktadır. İngilizce dilinde bilgisi olan kişilerin program satırlarına baktıklarında, her satır için ne olduğunu tahmin etmesi nispeten kolaydır.

İşte dijital bilgisayarların anladığı 0 ve 1'den oluşan makine dili ile İngilizce kökenli bir MIPS Assembly kodunu beraberce gösterildiği listeye bakalım.

MIPS Makine Dilindeki Program Kodu	**MIPS Assembly Dilindeki Program Kodu**
00000000 10100111 00011000 00100000	add $3, $5, $7
10101100 00100010 00000000 00000000	sw $2, 0($1)
00000011 11100000 00000000 00001000	jr $31

MIPS makine dili ve MIPS Assembly dilinde yazılmış aynı görevi işi yapan ikiprogram, sol sütununda gösterilen program kodu MIPS makine dilinde hazırlanmış olan ve sıfır ve birlerden oluşmakta olan program satırlarıdır. Şeklin sağ tarafında gösterilen ise aynı kodun MIPS Assembly dilindeki karşılığıdır.

Görülebileceği gibi insanlar açısından sevimsiz gözüken 0 ve 1'ler, Assembly dili kullanılması sayesinde programcının direkt olarak kullanmasına gerek kalmamaktadır.

ÖRNEK

```
1        add $2, $0, $0
2
3        add $3, $0, $0
4
5         addi $4, $0, 5
6
7
8        dongu: add $3, $2, $3
9
10        addi $2, $2, 1
```

Örnek program kodu .MIPS Assembly dilinde yazılmış 0'dan 4'e kadar olan sayıları bir döngü (loop)

MIPS Assembly kodunda toplam sekiz (8) işlem bulunmaktadır. Programın okunabilirliğini arttırmak için sol tarafta programın satır sayıları gösterilmektedir. Satırlardaki renklendirmeler özel bir anlam ifade etmemekte olup yalnızca okunabilirliği arttırmak için kullanılmıştır.
Örnekteki programın toplam sekiz işlemden oluştuğundan bahsettik. Oysaki şeklin sol tarafında gösterilen satır sayıları on yedi (17) satırdan oluşmaktadır. Buradaki fark bu örnek MIPS Assembly programında bazı "boş satırların bulunmasından" kaynaklanmaktadır. Daha önceki bölümde bahsedildiği gibi boş satırların program üzerinde bir etkisi bulunmamaktadır. Bu yüzden boş satırlar için bir işlem yapılmaz. Programın özelliklerini inceleyelim. Birinci satırda ilk işlem yapılmaktadır. Bu satırda "**add $2, $0, $0**" komutu ile yapılan işlem sıfır (0) numaralı yazmaç (register) ile yine sıfır (0) numaralı yazmacı toplamaktadır. Yapılan bu toplama işleminin sonucu ise iki (2) numaralı yazmaça (register) yüklemektedir.

Burada dikkat edilmesi gereken noktalardan birisi $0'ın (sıfırıncı yazmacın (register'ın)) değerinin sıfır (0) olmasıdır. MIPS sistem yapısına göre "*sıfırıncı yazmaç ($0) her zaman sıfır değerine sahiptir*". Buradan hareketle Şekil 4.1'deki programın birinci satırındaki komutta yapılan işlem sayesinde sistemdeki ikinci yazmaça sıfır değeri atanmaktadır. Bu işlemin çalışma şekli ve anlamsal yapısı Şekil 4.2'de gösterilmektedir.

MIPS Kodu	Anlamı	Aldığı Değer
add $2, $0, $0	$2 ← $0 + $0	$2 ← 0 + 0 Diğer bir değişle $2 ← 0

Şekil 4.2 add $2, $0, $0 komutunun özellikleri

MIPS Assembly programının ikinci komutu ise üçüncü satırda yer alan "**add $3, $0, $0**" işlemidir. Dikkat edilirse ikinci komut ikinci satırda değil üçüncü satırdadır. Bunun sebebi programı yazan kişinin bir satır boşluk bırakmayı tercih etmesinden kaynaklanmaktadır.
Programın ikinci komutunda yapılan işlem ilk işleme benzer şekilde bir toplama işlemidir. İlk işlemde olduğu gibi yine $0'ın değeri $0 ile toplanarak sonuç değer bu defa $3'e yazılmaktadır. Buradan hareketle programdaki ikinci işlemin çalışma şekli ve anlamsal yapısı Şekil 4.3'te görülmektedir.

MIPS Kodu	Anlamı	Aldığı Değer
add $3, $0, $0	$3 ← $0 + $0	$3 ← 0 + 0 Diğer bir değişle $3 ← 0

Şekil 4.3 add $3, $0, $0 komutunun özellikleri

MIPS Assembly programının üçüncü işlemi, beşinci satırda yer alan "**addi $4, $0, 5**" komutudur. Bu komut ile yapılan işlem ilk iki komutta olduğu gibi bir toplama işlemidir. Diğer taraftan bu komutta yapılan toplama işlemi bir yazmaç (register) ile doğrudan (immediate) sayı arasında yapılmaktadır. Dolayısı ile bu komut ile yapılan işlem 5 sayısı ile $0 yazmacının değerini toplayıp sonucu $4 yazmacına (register) yüklemektir.

Bu aşamada komuta dikkatli bakıldığında ilk iki komutta toplama işlemi yapılırken "**add**" işlemi kullanılmıştır. "add" işlemi iki yazmacı (register) toplarken kullanılan bir komuttur. Diğer taraftan bir yazmaç (register) ile bir doğrudan sayıyı toplamak istenilmesi durumunda "addi" komutu kullanılır. "addi" komutundaki "i" harfi doğrudan (immediate) bir sayı ile işlem yapılacağını göstermektedir.

MIPS Kodu	Anlamı	Aldığı Değer
addi $4, $0, 5	$4 ← $0 + 5	$4 ← 0 + 5 Diğer bir değişle $4 ← 5

Şekil 4.4 addi $4, $0, 5 komutunun özellikleri

MIPS programının dördüncü komutu sekizinci satırda yer alan "**dongu: add $3, $2, $3**" komutudur. Bu komutta önceki ilk üç komuttan farklı olarak "**dongu**" isminde bir kelime bulunmaktadır. Bu kelime MIPS Assembly kodunda "etiket (label)" işlevini görmektedir. Bir programda etiket kullanılması sayesinde yazılan programın herhangi bir komutundan etiketin bulunduğu bölüme erişim sağlanır.

Dolayısı ile Şekil 4.1'de gösterilen programda "dongu" etiketinin bulunduğu noktaya programın çalışması sırasında erişim mümkün olmaktadır. Bu programda "dongu" etiketinin kullanımı ile ilgili Şekil 4.1'deki programın ileriki komutlarındaki bağlantı görülecektir.
Şekil 4.1'deki programın dördüncü komutun diğer kısmında ise (add $3, $2, $3) toplama işlemi yapılmaktadır. Bu kısımda yapılan iki numaralı yazmaç ile (register) üç numaralı yazmacın değerini toplayıp, sonucu üç numaralı yazmaca yazmaktadır. Şekil 4.5'te sekizinci satırdaki dördüncü komutun çalışma yapısı görülmektedir.

MIPS Kodu	Anlamı	Aldığı Değer
dongu: add $3, $2, $3	$3 ← $2 + $3	Çalıştığı sırada $2 ve $3 yazmaçlarının değerlerinin toplamını $3 yazmaçına yazar. İlk çalışması sırasında aldığı değer: $3 ← 0 + 0

Şekil 4.5 add $3, $2, 3 komutunun özellikleri

Şekil 4.1'deki programın beşinci komutu ise onuncu satırda yer almaktadır. Burada "**addi $2, $2, 1**" işlemi ile 2 numaralı yazmacın değeri ile 1 sayısı toplanmaktadır. Bu işlemde "1" doğrudan bir sayı olduğundan, burada kullanılan toplama komutu olarak "addi" kullanılmaktadır. Dolayısı ile bu programdaki beşinci komutun çalışma yapısı Şekil 4.6'da gösterilmektedir.

MIPS Kodu	Anlamı	Aldığı Değer
addi $2, $2, 1	$2 ← $2 + 1	Çalıştığı sırada $2 yazmaçının değerleri ile 1 doğrudan sayısı toplamını $2 yazmaçına yazar. İlk çalışması sırasında aldığı değer: $2 ← 0 + 1

Şekil 4.6 addi $2, $2, 1 komutunun özellikleri

Şekil 4.1'deki programın altıncı komutu on ikinci satırda yer alan "**bne $2, $4, dongu**" komutudur. Bu komutta ikinci yazmaç ($2) ile dördüncü yazmaçların ($4) değerlerinin durumu kıyaslanmaktadır. Eğer bu iki yazmacın değerleri birbirinden farklı ise (değerleri eşit değilse) "dongu" ismindeki etiketinin (label) bulunduğu yere dallanma/atlama (branch) işlemi gerçekleştirilir. Bu sayede

iki yazmacın değeri eşit değilse programın işleyiş yönü sıradaki komut yerine komut içinde belirtilen etiketin bulunduğu yere yönlenmektedir. Diğer taraftan eğer "bne" komutundaki iki yazmacın değeri eşitse sıradaki komut ile programın çalışması devam etmektedir.

Buradan hareketle Şekil 4.1'deki MIPS programının altıncı komutunun çalışma yapısı Şekil 4.7'de gösterilmektedir.

MIPS Kodu	Anlamı	Aldığı Değer
bne $2, $4, dongu	Eğer $2 != $4 dongu ye git Diğer durumda Sıradaki komut	$2 ve $4 yazmaçlarının değerleri kıyaslanır. Eğer iki yazmaçın değeri birbirine eşit değilse "dongu" etiketine dallanma/atlama yapar. Diğer durumda ise (yani iki yazmaçın değeri birbirine eşitse) programda sıradaki komut ile çalışır. Programın ilk çalışması sırasında aldığı değer: 1!=5 olduğundan "dongu" etiketinin bulunduğu noktaya atlar

Şekil 4.7 bne $2, $4, dongu komutunun özellikleri

Programda yer alan altıncı komut olan "bne $2, $4, dongu" ile ilgili yapılan açıklamalara dikkat edilirse, programda sekizinci satırda yer alan dördüncü komutta yer alan "dongu" etiketine dallanma/atlama (branch) işlemi altıncı komut ile gerçekleştirilmektedir.
Diğer bir değişle, Şekil 4.1'de gösterilen MIPS Assembly programındaki dördüncü, beşinci ve altıncı komutlar arasında bir döngü oluşmaktadır. Bunun sebebi programın başında (üçüncü komutla) "5" değeri yüklenmiş olan dördüncü yazmaç ($4) ile başlangıçta değerine (birinci komutla) "0" yüklenmiş olan ikinci yazmacın ($2) değerleri eşitlenene kadar sürekli altıncı komuttan dördüncü komutun olduğu bölüme atlama/dallanma yapılmaktadır.

Bu bilgilerden yararlanarak program tekrar incelendiğinde programın yapısı şu şekildedir. Öncelikle "0" değerine sahip olan ikinci yazmacın değeri ($2) "dongu" etiketi ile "bne" komutu arasında (sekiz ve on ikinci satırlar arasında) oluşturulmuş bir döngü (loop) oluşturulmaktadır. Oluşturulan bu döngüden yararlanılarak programda başlangıçta "0" değerine sahip olan ikinci yazmacın ($2) her döngü tekrarı sırasında değeri bir artmaktadır ($2 ← $2 + 1). Bu durumda oluşturulan döngüde ikinci yazmacın ($2) değeri "5" e ulaşana kadar bu döngü devam etmektedir. Bunun sebebi dördüncü yazmacın ($4) "5" değerine sahip olduğundan oluşturulan döngü iki yazmacın eşit olduğu "5" değerinde devre dışına çıkacaktır.

Oluşturulan bu döngüdeki yapıyı incelemeye devam edildiğinde üçüncü yazmacın ($3) bir tür toplama işlemi yaptığı görülmektedir. Üçüncü yazmacın üzerine yapılan bu toplama işlemindeki amacın "0" dan "4" e kadar olan sayıların toplamını vermektedir.
Bu kısımda anlatılanları toparladığımızda ikinci yazmaç ($2) sayaç (counter) görevi görmekte, üçüncü yazmaç ($3) ise sayıların toplamını sırayla biriktirmektedir.

Daha önce belirtildiği gibi, ikinci yazmacın ($2) değeri ile dördüncü yazmacın değeri birbirine eşitlendiğinde ($2 5 değerine ulaşınca) oluşturulan döngüden çıkılmaktadır. Bu aşamaya gelindiğinde program sıradaki komut ile devam etmektedir (atlama olmaz, aşağıdaki komuta geçer).

Şekil 4.1'deki on beşinci satırda yer alan yedinci komut ise "**sw $3, 108($0)**" işlemidir. Bu komutta yapılan işlem üçüncü yazmacın değerini hafızaya yazılma işlemidir. Burada hafızanın hangi bölümüne yazılacağı hesabında "108 ile sıfırıncı yazmacın ($0)" değerlerinin toplamı ile gösterilen bölüm kullanılmaktadır. Diğer bir değişle, üçüncü yazmacın içinde bulunan "10" değeri (neden 10 biraz düşünelim) hafıza üzerinde "108 + 0" ile gösterilen bölüme yazılmaktadır.

Buradan hareketle Şekil 4.1'deki programdaki yedinci komutun özellikleri Şekil 4.8'de gösterilmektedir.

MIPS Kodu	Anlamı	Aldığı Değer
sw $3, 108($0)	Hafıza[108 + $0] ← $3	Hafıza[108 + 0] ← 10

Şekil 4.8 sw $3, 108($0) komutunun özellikleri

Bu arada eğer halen yedinci komut itibarı ile üçüncü yazmaçta neden 10 değeri olduğunu düşünüyorsanız, bunun sebebi 0'dan 4'e kadar olan sayıların toplamının 10 olmasından kaynaklanmaktadır.

Şekil 4.1'de gösterilen programın sekizinci ve son komutu "**jr $31**" işlemidir. On yedinci satırda yer alan bu komutta otuz bir numaralı yazmacın ($31) gösterdiği adrese "atlama (jump)" işlemi yapılmaktadır. "jr" komutu koşulsuz bir atlama işlemidir. Bu yüzden programda komut çalıştırıldığında otomatik olarak yazmacın içinde gösterilen yere atlanır (jump).

"jr $31" komutunun bir özel yapısı ise "jr" komutunda otuz birinci yazmaç ($31) kullanılması durumunda programdan çıkılıp işlemler işletim sistemine (operating system) devredilmektedir. Bu özelliğinden ötürü, MIPS Assembly programları sık sık "jr $31" komutu ile bitmektedir.

Şekil 4.1'de gösterilen MIPS Assembly programının sekizinci ve son komutu hakkındaki özellikler Şekil 4.9'da görülmektedir.

MIPS Kodu	Anlamı	Aldığı Değer
jr $31	Sıradaki işlem ← $31	Programda çalıştırılacak olan sıradaki işlem otuz birinci yazmaçın ($31) göstermiş olduğu adrestir

Şekil 4.9 jr $31 komutunun özellikleri

SORULAR

1. MIPS Assembly dilinde "add" komutu ne işe yaramaktadır?
a. Belleği sıfırlamaya
b. Yazmaça ad (isim) vermeye
c. Bilgisayardan bip sesi çaldırmaya
d. Toplama işlemi yapmaya
e. Kafadan atma işlemi yapmaya (at)

2. MIPS Assembly dilinde "sw" komutu ne işe yaramaktadır?
a. Toplama işlemi yapmaya
b. Hafızaya bilgi yüklemeye (memory ye)
c. Hafızayı Silip, Vakumlamaya
d. Hafızadan bilgi yüklemeye (memory den)
e. Çıkarma işlemi yapmaya

4.9.MIPS ASSEMBLY SİMÜLATÖRÜ

MIPS Assembly dili RISC sistem modeli kullanan bilgisayar sistemlerinden birisi olan MIPS bilgisayar sistemleri için kullanılmaktadır. Diğer taraftan, Intel 80x86 temelli başka bilgisayar sistemleri kullanan programcılarında MIPS Assembly programını kullanabilmesi açısından simülatör aracılığı ile kullanılabilmektedir.
Bu amaca yönelik olarak hazırlanmış programlardan birisi "SPIM" simülatörüdür. İlk sürümü Ocak 1990 itibarı ile tasarlanmış olan bu simülasyon sürekli yeni sürümler ile güncellenmektedir. Bu simülatör Unix, Linux ve Microsoft Windows işletim sistemleri için kullanılabilmektedir.
SPIM simülatörü sayesinde MIPS Assembly programlama yapmak direkt olarak MIPS sistemi üzerinde Assembly programı yazmak ve düzeltmekten de daha kolaydır. Bunun sebebi, programcıya daha fazla açıklayıcı bilgi verebilmekte ve program yapmayı kolaylaştırmaktadır.
Değişik versiyonları olan SPIM simülatörünün başlıca ürünleri "*console spim*", "*Xwindows spim*" ve "*Microsoft Windows spim*" gruplarından oluşmaktadır.
"*Console spim*" SPIM simülatörünün komut satırı üzerinden çalıştırılarak kullanılan ürünüdür.
"*Xwindows spim*" SPIM simülatörünün Unix türevi işletim sistemlerindeki Xwindows üzerinde kullanılan ürünüdür.

"*Microsoft Windows spim*" ise SPIM simülatörünün Microsoft Windows üzerinde kullanılan bir üründür.

4.10. MIPS ASSEMBLY İLE C DİLİ KIYASLAMASI

MIPS Assembly programlama dili kullanılarak MIPS bilgisayar yapısı kullanan bir sistemde direk olarak programlama yapılabilmektedir. Bu kısımda Bilgisayar Biliminde yaygın olarak kullanılan C programlama dili ile MIPS Assembly dili arasındaki çeşitli farklar aşağıda gösterilmektedir.

Özellik	MIPS Assembly Programlama Dilindeki Durumu	C Programlama Dilindeki Durumu
Aritmetik işlemler	add, addi, sub, mult, slt	+, -, *, %, ++, --, <, >,
Hafızaya erişim	lw, sw, lb, sb, lh, sh	k, *k, k[i], k[i][j]
Kontrol	beq, bqez, bgezal, bgtz, blez, bltz, bltzal, bne, j, jal, jr, syscall	if-else, while, do-while, for, switch, procedure call, return
Veri Tipleri	byte (8 bit), yarım-word (16 bit), word (32 bit), single floating point (32 bit), double floating point (64 bit)	int, short, char, unsigned, float, double, pointer ve diğer veri tipleri
Yazılan programın bilgisayarın anladığı makine diline çevrim süreci	Yazılan MIPS Assembly programı "Assembler" kısmına girer. Assembler programı makine dilindeki programı oluşturur	Yazılan C programı öncelikle "Derleyici (Compiler)" kısmına girer. Derleyici kısmından Assembly programı çıkmaktadır. Daha sonra bu program makine diline çevrilmek için Assembler'a girer

SORULAR

1. MIPS sisteminde sıfırıncı yazmacın (register zero) ($0) değeri kaçtır?
a. Sabit bir değeri yoktur, programın o anki durumuna göre değişir
b. İki üssü sıfırdan dolayı = "birdir (1)"
c. Sıfırdır (0)
d. Beşinci yazmacın değerinin yarısıdır ($5/2)
e. Otuz ikidir (32)

2. MIPS Assembly programlama dilinde "addi" komutunda yer alan "i" harfi neyi temsil etmektedir?
a. Komutta kullanılacak olan yazmaçların ilk (initial) değer alacağını
b. Komutun "iyi" olduğunu
c. Komutun belirteçi (indicator) olduğunu
d. Komutta doğrudan (immediate) sayı işlemi olduğunu
e. Komutta örnek nesne (instantiate) oluşturma olduğunu

3. Assembly programında etiket (label) nedir?
a. Programın toplam satır sayısını gösterir
b. Programı yazan kişinin ürününe biçtiği bedeli belirten etiket
c. Her programın başında mutlak bir tane bulunmaktadır
d. Her programın sonuna mutlak bir tane bulunmaktadır
e. Program içinde belirli bir noktaya ulaşılmasını sağlayan bir tür yer belirteci

4. MIPS Assembly dilinde "jr" ne işe yaramaktadır?
a. Kıyaslama işlemi yapar
b. Çarpma işlemi yapar
c. Toplama işlemi yapar
d. Programı yazmacın (register) gösterdiği adrese atlamasını (jump) sağlar
e. Programı kıyaslama işlemi yaptıktan sonra atlamasını sağlar

5. MIPS Assembly dilinde "bne" ne işe yaramaktadır?
a. Yazmaça doğrudan bir sayı ekler
b. Çarpma işlemi yapar
c. Toplama işlemi yapar
d. Programı kıyaslama işlemi yapmadan gösterilen adrese dallanma/atlama (branch) yapar
e. Programı kıyaslama işlemi yaptıktan sonra gösterilen adrese dallanma/atlama (branch) yapar.

****

4.11. CISC MİMARİSİNE GİRİŞ

İngilizce bir kısaltma olan CISC "*Complex Instruction Set Computer*" kelimelerinin baş harflerinden oluşmaktadır. Bu kelimeler Türkçeye "*Karmaşık Komut Kümesi Bilgisayarı*" şeklinde çevrilebilecektir.

CISC mimarisinin isminden de anlaşılabilecek özelliklerinden birisi sistemde kullanılan komutlar karmaşık (complex) yapıdadır. Karmaşık yapıdaki bu komutların gerek uzunlukları farklı olmakta, gerekse de çalışma süreleri birbirinden önemli ölçüde farklılık göstermektedir.
CISC mimarisindeki karmaşıklık donanımda yer almaktadır. Karmaşık komutlar kullanması ile birlikte sistemin donanım özelliği de karmaşık bir yapıdadır. CISC mimarisindeki özelliklerden birisi komutların uzunluğunun değişik yapıda olmasıdır. İhtiyaç duyulan komut ihtiyaç duyulduğu uzunlukta bir yapıya sahip olacak şekilde tasarlanabilir.

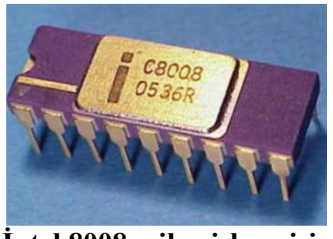
Uzunluğu değişen yapısı ile CISC mimarisindeki komutların hangi türde olduğunu tespit etmek daha zor hale gelmiştir. Diğer bir değişle, komutların hangi komut olduğunun belirlendiği "opcode" kısmının anlaşılması oldukça zordur.

CISC mimarisinin diğer bir özelliği ise hafızadan yazmaça yükleme ve yazmaçtan hafızaya yükleme işlemleri aynı işlem ile gerçekleştirilebilmektedir.

Bu mimarideki komutların karmaşık yapısı ve aynı komut üzerinde birden fazla işlemin yapılabilmesi sayesinde CISC mimarisinde yazılacak programlarda az komut bulunması yeterli olabilmektedir.

CISC mimarisinde pipelining işlemini gerçekleştirebilmek için eş zamanlı olarak komutların küçük parçalara bölümlenerek işlemcide o şekli ile çalıştırılması gerekmektedir.

4.11.1. CISC Mimarisi Kullanan Örnek Sistemler

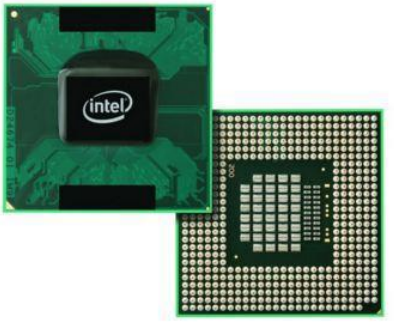


İntel 8008 mikroişlemcisi

AMD 29000

Cyrix 486SLC


Hp firmasının kullandığı yeni nesil mikroişlemci türü.

Intel Core 2 Duo T9900 mikroişlemci kullanan Dell firmasının Vostro 1220 model dizüstü bilgisayarı (laptop)


Intel Xeon 5500 mikroişlemci kullanan Fujitsu firmasının Primergy TX300 S5 iş istasyonu (workstation)

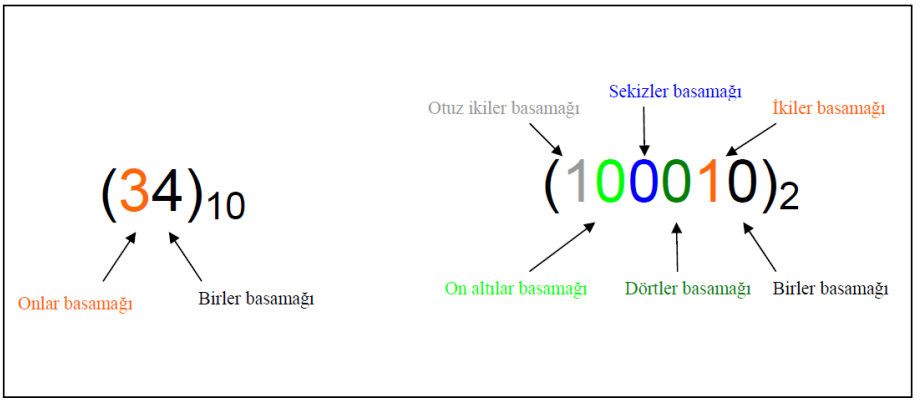
Intel Core 2 Duo mikroişlemci kullanan Apple firmasının iMac bilgisayarı

4.11.2. CISC ve RISC MİMARİSİNİN KARŞILAŞTIRILMASI

Özellik	CISC Mimarisi	RISC Mimarisi
Komut Yapısı	Karmaşık	Basit
Komut Uzunluğu	Değişken	Sabit
Komut Çalışma süresi	Çok Değişken	Birbirine yakın
Donanım Yapısı	Karmaşık	Basit
Yazılım Maliyetleri	Daha az	Daha çok
Hafıza işlemleri	Aynı komut ile yazmaçtan hafızaya ve hafızadan yazmaça işlem yapılabilir	Yazmaçtan hafızaya ayrı, hafızadan yazmaça ayrı komut ile bilgi aktarma yapılmaktadır.
Komutların Anlaşılması (decode edilmesi)	Komutların değişken yapıda olmasından dolayı daha zordur	Komutların benzer yapıda olmasından dolayı daha kolaydır
Yazılan Assembly programı kodunun uzunluğu	Daha az sayıda program kodu kullanılır	Daha çok sayıda program kodu kullanılır
Pipeline Kullanımı	Bu işlem için komutlar küçük alt komutlara bölünmeli	Pipeline kullanımına hazır
Sistemde Kullanılan Hafıza	Normal	Daha Fazla
Sistemdeki Yazmaç Sayısı	Az	Çok
Sistemde Tanımlı Komut Sayısı	Çok	Az
Sistemde Tanımlı Komut Listesinden Kullanılan Komut Yoğunluğu	Tanımlı komutların dörtte ya da beşte biri gibi küçük bir kısmı kullanılmaktadır.	Tanımlı komutların çoğu kullanılmaktadır.
Günümüzde Sıklıkla Kullanıldığı Sistemler	Masaüstü, işistasyonu, sunucu ve dizüstü (laptop/notebook) bilgisayarlar	Oyun konsolları, cep telefonları ve gömülü bilgisayar sistemleri

Kullanıcılar bu gözlemlerden yararlanarak günümüzde RISC sistemlerini daha çok oyun konsolları, cep telefonu ve gömülü bilgisayar sistemlerinde kullanmaktadır. Bununla birlikte halen RISC mimarisi kullanan işistasyonları ve sunucu sistemleri de bulunmaktadır.

Diğer taraftan CISC sistemleri ise günümüzde genellikle masaüstü bilgisayar, iş istasyonu, dizüstü (laptop/notebook) ve sunucu gibi bilgisayarlarda bulunmaktadır.



Onluk tabanında bir sistem:

$34 = (34)_{10} = 3*10 + 4*1$

$100010 = (100010)_2 = (1*32) + (0*16) + (0*8) + (0*4) + (1*2) + (0*1)$

4.11.3. CISC MİMARİSİNDE YAZMAÇLAR

CISC mimarisinde yer alan 8086 yapısında aşağıdaki yazmaçlar (registers) bulunmaktadır.

Genel amaçlı yazmaçlar (registers):

AX: Accumulatör yazmacıdır. AH ve AL olmak üzere sekiz (8) bitlik iki kısımdan oluşur.
BX: Temel adres yazmacıdır. BH ve BL olmak üzere sekiz (8) bitlik iki kısımdan oluşur.
CX: Count (sayma) yaçmaçıdır. CH ve CL olmak üzere sekiz (8) bitlik iki kısımdan oluşur.
DX: Data (veri) yazmacıdır. DH ve DL olmak üzere sekiz (8) bitlik iki kısımdan oluşur.

Pointer amaçlı yazmaçlar (registers):

SP (Stack Pointer yazmacı): Stackın en üst noktasını göstermektedir.
BP (Base Pointer yazmacı): Adres ayarlamaları ile ilgili kullanılır.

Index yazmaçları (registers):
SI (Source Index, Kaynak İndeksi),
DI (Destination Index, Hedef İndeksi)

Segment yazmaçları (registers):
CS (Code Segment, Kod Segmenti)
DS (Data Segment, Veri Segmenti)
SS (Stack Segment, Stack Segmenti)
ES (Extra Segment)

Program Counter yazmacı (register):
IP (Instruction Pointer, Komut İşaretleyicisi)

Flag yazmacı (register):
FR (Flag Register, bayrak yazmacı). Bu yazmacın
CF (Carry Flag, Taşma Bayrağı (Belirteci))
PF (Parity Flag, Eşlik Bayrağı (Belirteci))
AF (Auxiliary Flag, Yardımcı Bayrağı (Belirteci))
ZF (Zero Flag, Sıfır Bayrağı (Belirteci))
SF (Sign Flag, İşaret Bayrağı (Belirteci))
TF (Trap Flag, Yakalama Bayrağı (Belirteci))
IF (Interrupt Flag, Kesme Bayrağı (Belirteci))
DF (Direction Flag, Yön Bayrağı (Belirteci))
OF (Overflow Flag, Taşma Bayrağı (Belirteci))

SORULAR

1. İngilizce bir kısaltma olan "*CISC* kelimesinin Türkçe açılımı" nedir?
a. Kalabalık Belirteç Kümesi Bilgisayarı
b. Karakter İçeren Bilim Bilgisayarı
c. Karmaşık Komut Kümesi Bilgisayarı
d. Kaynak İletişim Sözcük Bilgisayarı
e. Komut İçeren Saat Bilgisayarı

2. İngilizce bir kısaltma olan "*CISC* kelimesinin İngilizce açılımı" nedir?
a. Condensed Indexed Science Computer
b. Complex Instruction Set Computer
c. Condensed Instruction Set Compiler
d. Complex Indication Science Computer
e. Compiler Instruction Science Computer

3. Aşağıdakilerden hangisi doğrudur?
a. Intel 8008 mikroişlemcisi RISC mimarisinde yer alır
b. CISC mimarisi genellikle cep telefonlarında kullanılır
c. RISC mimarisinde sistemde tanımlı komut sayısı azdır
d. RISC mimarisinin donanım yapısı karmaşıktır
e. RISC mimarisinin komut yapısı karmaşıktır

4. Aşağıdakilerden hangisi doğrudur?
a. CISC mimarisinin komut uzunluğu sabittir
b. CISC mimarisinin yazmaç sayısı çoktur
c. CISC mimarisinde komutların anlaşılması daha kolaydır
d. CISC mimarisinde komut yapısı karmaşıktır
e. CISC mimarisinde yazılan Assembly programı kodunun uzunluğu daha çoktur

5. Aşağıdakilerden hangisi bir CISC mimarisi kullanan sistem değildir?
a. HP Pavillion, Intel Core 2 Quad Q8200
b. Dell Vostro 1220, Intel Core 2 Duo T9900
c. Playstation 3, Cell Broadband Engine, 3.2 GHz
d. Acer Ferrari, AMD ATH X2
e. Apple iMac, Intel Core 2 Duo

4.11.4. SEGMENT YAPILARI ve ADRESLEME YÖNTEMLERİ

Bu hafta mikrobilgisayar sistemlerinin yapısal iki temel grubundan birisi olan "CISC *(**C**omplex **I**nstructions **S**et **C**omputer)*" mimarisinde segment yapıları ve adresleme yöntemleri ile ilgili bilgiler verilecektir.

4.11.5. CISC ASSEMBLY PROGRAMLAMADA SEGMENT YAPISI

CISC Assembly programlama yapısında segmentler kullanılmaktadır. Bir assembly programının üç segmenti bulunmaktadır. Bunlar "DATA (veri) segment", "CODE (kod) segment" ve "STACK segment" tir.
Bu kısımda CISC mimarisinin bu üç segmenti hakkında bilgiler verilecektir.

DATA Segment

Bu bölümde veri (data) ile ilgili bilgiler tanımlanmaktadır. Bu bölümde tanımlanan veriler yazılan assembly programı boyunca kullanılabilecek verileri göstermektedir.

ORTASAYI DB 17H

DATA segment satırında ORTASAYI isminde ve bir (1) byte büyüklüğünde bir yeri hafızada tanımlanmıştır.

"DB" İngilizce "define byte" kelimelerin baş harflerinden gelmekte olup, Türkçe byte tanımlama anlamına gelmektedir.

17H ise, tanımlanan hafıza bölümüne 17H değeri verilmiş olmaktadır.

SAYI DW 20H

Data segment bölümünde hafızada 20H değeri olan bir wordlük alan ayrılmaktadır. "DW" İngilizce "define word" kelimelerin baş harflerinden gelmekte olup, Türkçe word tanımlama anlamına gelmektedir.

SONSAYI DW ?

Data segment bölümünde hafızada bir wordlük bir yer ayrılmaktadır. Bu word verisine ilk değer verilmemektedir. "?" sembolü yeni tanımlanan SONSAYI isimli veriye ilk değer verilmediğini göstermektedir.

STACK Segment

Yazılan Assembly programı için stack segmentte bir yer ayrılan bölümdür.

.STACK 128

Hafızada 128 bytelık bir alanı stack için ayrılmasına yarayan satır.

CODE Segment

CODE segmentin amacı programın kod satırlarını göstermektir. Her bir satıra bir komut gelecek şekilde organize edilen bu segmentte mov, add, int gibi komutlar yer almaktadır. Örnek bir CODE segment başlangıcı gösterilmektedir.

.CODE

4.11.6. ASSEMBLY PROGRAMI MODEL TÜRLERİ

Yapılan hafıza modeli seçimi yazılan assembly programının hafızayı kullanma şeklini belirlemektedir. Kullanılabilecek hafıza modelleri arasında "TINY", "SMALL", "COMPACT", "MEDIUM", "LARGE" ve "HUGE" bulunmaktadır.
Bu modellerden TINY olanı .com uzantılı dosyalarda kullanılmaktadır. İngilizce'de tiny kelimesi çok küçük anlamına gelmektedir. com uzantılı dosyalar hafızada az yer kapladığı için TINY model hafıza yapılanmasına ihtiyaç duymaktadır.
SMALL model .EXE programları için kullanılan modellerin başında gelmektedir.

.MODEL TINY

ÖRNEK UYGULAMA

```
.MODEL SMALL
.STACK 64
.DATA
SAYI DB 20H
MAKS DB 50H
FARK DB 10H
.CODE
MAIN PROC FAR
MOV AX, @DATA
MOV DS, AX
MOV AL, SAYI
MOV AH, MAKS
SUB AH, AL
MOV FARK, AH
MOV AH, 4CH
INT 21H
MAIN ENDP
END MAIN
```

SMALL modelinde bir program olduğu görülmektedir. STACK segment belirlenmiş ve 64 bytelık bir bölüm ayrılmıştır. DATA segmentte "SAYI", "MAKS" ve FARK isminde üç değişken tanımlanmıştır. Byte olan bu değişkenlerin değerleri sırası ile 20H, 50H ve 10H dir.

CODE segment MAIN PROC FAR ile başlamakta, daha sonra "MOV, AX, @DATA" ve "MOV DS, AX" komutları ile devam etmektedir. Arka arkaya gelen iki mov komutu ile DATA segmenti ilkdeğeri verme (initialization) gerçekleşmektedir. Bunun sebebi DOS işletim sistemi DATA segment için bir

ilk değer verme işlemi yapmamasıdır. Diğer taraftan DOS işletim sistemi CODE segment ve STACK segment için ilk değer verme işlemi gerçekleştirmektedir. CODE segmentin initialization işleminde AX yazmacı (register) kullanılmaktadır.
Programın esas işlem yapılan dört komutluk kısmında ise şunlar yapılmaktadır. Öncelikle SAYI isimli değişkenin değeri AL yazmacına "MOV AL, SAYI" komutu ile yazılmaktadır. Daha sonra MAKS değeri AH yazmacına "MOV AH, MAKS" komutu ile yazılmaktadır. Bu iki yazmacın arasındaki fark ise "SUB AH, AL" ile alınmaktadır. AH yazmacında tutulan sonuç ise "MOV FARK, AH" komutu ile FARK değişkenine yazılmaktadır.
Bu sayede yukarıda gösterilen dört satır ile iki değişkenin değerinin çıkartması yapılmıştır. Sadece bir çıkartma işleminin dört (4) satır sürmesinin sebebi Assembly programlamada doğrudan hafıza üzerinden işlem yapılamamasıdır. Örneğin burada olduğu gibi iki değişkenin farkı alınacaksa (ya da değişkenler toplanacaksa) öncelikle değişkenleri hafızadan yazmaçlara (register) yüklenmesi gerekmektedir. Değişkenler üzerinde yapılacak işlemler daha sonra gerçekleştirilmektedir.
Assembly programlarının yüksek seviyeli programlama dillerinden (high level programming languages) daha uzun yapıda olmasının nedenlerinden birisi budur.
Programın en altında bulunan "MOV AH, 4CH" ve "INT 21H" komutları programı sonlandırmak için kullanılmaktadır. Bu satırlardan sonra programın işleyişi DOS işletim sistemine geçip program sonlandırılmaktadır.
En altta yer alan "MAIN ENDP" ana prosedürün bittiğini göstermektedir. "END MAIN" komutu ise programın sonlandığını göstermektedir.

4.11.7. CISC ASSEMBLY PROGRAMLAMA KOMUT YAPISI ve MOV KOMUTU ÖZELLİKLERİ

Mikrobilgisayar sistemlerinin yapısal iki temel grubundan birisi olan "CISC ***(Complex Instructions Set Computer)***" mimarisinde Assembly programlamanın komut yapısı hakkında bilgiler verilecektir. Ayrıca, mov komutunun özellikleri hakkında çeşitli örneklerden bahsedilecektir.

4.11.8. CISC ASSEMBLY PROGRAMLAMADA KOMUT YAPISI

CISC Assembly programlama RISC mimarisinde olduğu şekli ile her komutun bir satıra yazılması ile gerçekleşmektedir.

etiket : işlem ; yorum

CISC mimarisi Assembly satırı üç bölümden oluşabilmektedir. Bunlardan "*etiket* (*label*)" kısmı bir Assembly programı içerisinden direkt olarak ulaşabilinecek (atlanabilinecek) yeri belirtmek için kullanılmaktadır. Bir satırın etiket (label) kısmına sahip olup olmaması tamamen isteğe bağlı bir durumdur. Bir CISC Assembly satırında etiketin (label) bitiminde "*iki nokta üst üste işareti* (:)" bulunması gerekmektedir.
CISC mimarisindeki Assembly programlarının komut satırında yer alan "*işlem* (*operation*)" kısmı ise komutun gerçek anlamda işlem yapılan yeridir. Bu kısımda aritmetik işlemler, mantık işlemleri, yazmaç arası bilgi taşıma, yazmaç ile hafıza arası bilgi aktarma ve diğer işlemler yapılmaktadır. Bir CISC Assembly komut satırında "*işlem* (*operation*)" kısmı da boş bırakılabilir. Diğer bir değişle, işlem yapılmayan bir satır olabileceği gibi, bir satırda yapılmak istenen komut yer alabilir.

CISC Assembly programı satır yapısında yer alan üçüncü ve son bölüm "*yorum* (*comment*)" kısmıdır. Bu kısımda yazılmakta olan programın gerek yazma ve gerekse hataları ayıklama (debugging) kısmında yapılmak istenen işlemleri hatırlamaya yardımcı olunması için kullanılan bir kısımdır.

ÖRNEK

```
.MODEL SMALL
.DATA
SAYI DB ?
.CODE
```

```
mov ax,@data ; Veri segmentinin adresi ax yazmacına yazılır
mov ds, ax ; ax yazmacındaki değer veri segment yazmacına
ara : mov ch, SAYI ; SAYI değişkenin değeri sekiz bitlik ch yazmacına
mov cl, ch ; ch yazmacının değeri cl yazmacına yazılıyor
mov ah, 4ch ; programı sonlandırmak için
int 21h ; kontrolü işletim sistemine bırakma
```

CISC Assembly Programlamada MOV İşlemi

CISC Assembly programlamada önemli işlemlerden birisi "*mov*" işlemidir. Bu işlemde yapılan uygulama kaynak olarak belirtilen yerden, hedef olarak belirtilen bölüme doğru bir yazma işlemi gerçekleştirilmesidir. MOV işleminde diğer bir değişle bir noktadaki bilgilerin diğer noktaya kopyalaması işlemi gerçekleştirilmektedir.

mov bl, 22h

CISC mimarisi için yazılan Assembly programında sekiz (8) bitlik bl yazmacına 22h(hexadecimal, onaltılık sayı sistemi) sayısı yüklenmesi

mov bl, cl

mov bl, bh

mov bx, cx

mov ebx, eax

32 bitlik iki yazmaç arasında yapılan kopyalama işlemi gösterilmektedir.

mov [1590], ax

Bir yazmaçtan, 1590 ile başlayan hafıza bölümüne yapılan kopyalama işlemi gösterilmektedir.

mov bx, eax

CISC mimarisi için yazılan Assembly programında 16 bitlik bir yazmaça, 32 bitlik bir yazmacın değeri yazılması mümkün değildir.

mov [2400], [1220]

Hafızanın 1220 kısmında bulunan değer hafızanın 2400 kısmına yazılmak istenmiştir. Diğer taraftan, CISC Assembly programlamada bir mov işleminde iki hafıza noktası arasında kopyalama yapılması mümkün değildir.

mov ax, 63459h

CISC mimarisi için yazılan Assembly programında on altı (16) bitlik ax yazmacına on altı (16) bitten daha fazla yer kaplayan bir doğrudan sayı (immediate) ataması yapılmak istenmiştir. Bu uygulamada boyut uyuşmazlığından dolayı uygun değildir.

mov 6345h, cx

On altı (16) bitlik cx yazmacının değeri on altı (16) bit yer kaplayan bir doğrudan sayıya (immediate) yazılmak istenmiştir. Doğrudan sayının (immediate) içine kopyalama yapılamayacağı için bu uygulama da uygun değildir.

KAYNAKLAR

1- **Lojik Devreler, Prof. Dr. Bekir ÇAKIR, Yrd. Doç. Dr. Ersoy BEŞER, Yrd. Doç. Dr. Esra KANDEMİR BEŞER**
2- **MEGEP- Elektrik Elektronik Teknolojisi, Lojik Uygulamaları I, Ankara 2007**
3- **Mikro Bilgisayar Sistemleri ve Assembler, Yrd. Doç. Dr. Cenk ATLIĞ, Trakya Üniversitesi, Tunca Meslek Yüksekokulu, Edirne 2010**