MİKROİŞLEMCİLER

Dünyanın ilk mikroişlemcisi **Intel 4004**, bir çip üzerinde **4** bitlik mikroişlemci tarafından programlanabilir bir denetleyiciydi.

Bit: 1 veya 0 her biridir.

Mikroişlemci (microprocessor): Mikroişlemci tabanlı bilgisayar sisteminin kalbinde mikroişlemci entegre edilmiştir. Bazen CPU (merkezi işlem birimi=**C**entral **P**rocessing **U**nit) olarak adlandırılan mikroişlemci, bilgisayar sistemindeki kontrol elemanıdır.

Mikroişlemci veriyolu ve G / Ç' yi (I/O) veri yolu adı verilen bir dizi bağlantı aracılığıyla kontrol eder. Bus veri yolu bir G / Ç veya bellek cihazı seçer, bir G / Ç cihazı veya bellek ile mikroişlemci arasında veri aktarır ve G / Ç ve belleği kontrol eder.. Bellek ve G / Ç, bellekte saklanan talimatlar ve mikroişlemci tarafından yürütülür.

Mikroişlemci bilgisayar sistemi için üç ana görevi yerine getirir:

(1) kendisi ve bellek veya G / Ç sistemleri arasında veri aktarımı,

(2) basit aritmetik ve mantık işlemleri,

(3) basit kararlar yoluyla program akışı.

Bunlar basit görevler olmasına rağmen, onlar aracılığıyla mikroişlemci neredeyse her türlü işlem veya görevi gerçekleştirir. Bu işlemler çok basittir, ancak onlar aracılığıyla çok karmaşık problemler çözülür.

Basit Aritmetik ve Mantıksal İşlemler (Simple Arithmetic and Logic Operations)

Operation	*Comment*
Addition	
Subtraction	
Multiplication	
Division	
AND	Logical multiplication
OR	Logic addition
NOT	Logical inversion
NEG	Arithmetic inversion
Shift	
Rotate	

Karar İşlemleri (Decision Operations)

Decision	*Comment*
Zero	Test a number for zero or not-zero
Sign	Test a number for positive or negative
Carry	Test for a carry after addition or a borrow after subtraction
Parity	Test a number for an even or an odd number of ones
Overflow	Test for an overflow that indicates an invalid result after a signed addition or a signed subtraction

Veriler bellek sisteminden veya dahili kayıtlardan çalıştırılır. Veri genişlikleri değişkendir.

ve bir byte (8 bit), word (16 bit) ve doubleword (32 bit) ,quadwords (64 bit) ve octalwords (128 bit).

byte -> **8** bit

Word ->**16** bit

Dword ->**32** bit

Quadwords -> 64 bit

Octalwords -> 128 bit

Veri Yolları (Buses): Veri yolu, bir bilgisayar sistemindeki bileşenleri birbirine bağlayan yaygın bir kablo grubudur. Bir bilgisayar sisteminin bölümlerini birbirine bağlayan veri yolları, adres, veri (data) ve(control) kontrol aktarımını mikroişlemci ile mikroişlemci belleği ve G / Ç sistemleri arasındaki bilgiler aktarımını sağlayan yollardır.

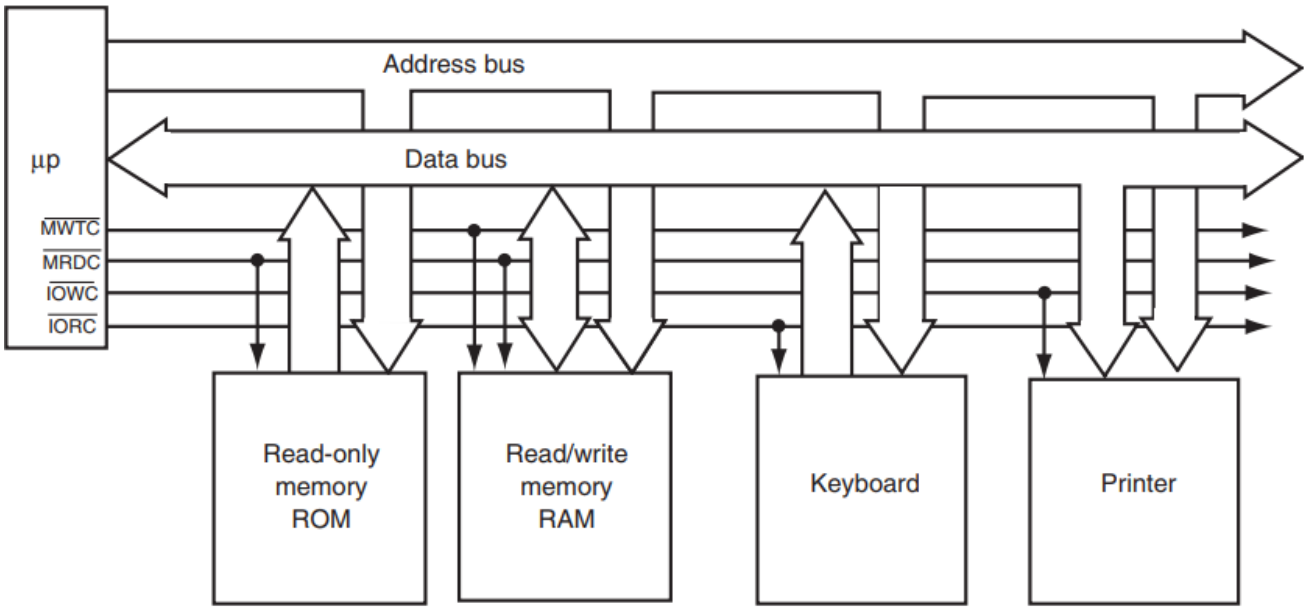
Mikroişlemci tabanlı bilgisayar sisteminde, bu bilgi aktarımı için üç veri yolu vardır:

1. adres,

2. veri,

3. kontrol

16 bit veri yolu adresleme 0000**H** 'dan başlar FFFF**H** biter. H sayının hegzadecimal yani 16 lık sayı sistemi olduğunu gösterir .32 bit veri yolu adresleme 00000000**H**–FFFFFFFF**H** aralığındadır. Aşağıdaki resimde veri yolları gösterilmektedir.



SAYI SİSTEMLERİ

Mikroişlemcinin kullanımı iyi anlayabilmek için **binary** (ikili), decimal (onluk) ve **hexadecimal** (onaltılı) sayı sistemleri hakkında bilgi sahibi olmalıyız.

Digit (rakam) : Sayılar bir sayı tabanından diğerine dönüştürülmeden önce bir sayı sisteminin rakamları anlaşılmalıdır.

Onluk ya da on tabanında (taban 10) bir sayının 0 ile 9 arası rakamlarla oluşturulur. Herhangi bir numaralandırma sistemindeki ilk basamak her zaman sıfırdır.

Örneğin, bir taban 8 (sekizli) sayı 8 rakam içerir: 0 ila 7; bir taban 2 (ikili) sayı 2 rakam içerir: 0 ve 1. Bir sayının tabanı 10'u aşarsa, ek basamaklar A ile başlayan alfabenin harflerini kullanır.

Hexadecimal Digit	*BCH Code*	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

ASCII (American Standard Code for Information Interchange)

ASCII (Amerikan Bilgi Değişimi için Standart Kod) verileri, bir bilgisayar sisteminin belleğindeki alfasayısal karakterleri temsil eder. Standart ASCII kodu, bazı eski sistemlerde pariteyi tutmak için kullanılan sekizinci ve en önemli bit olan toplam 7 bitlik bir koddur.

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

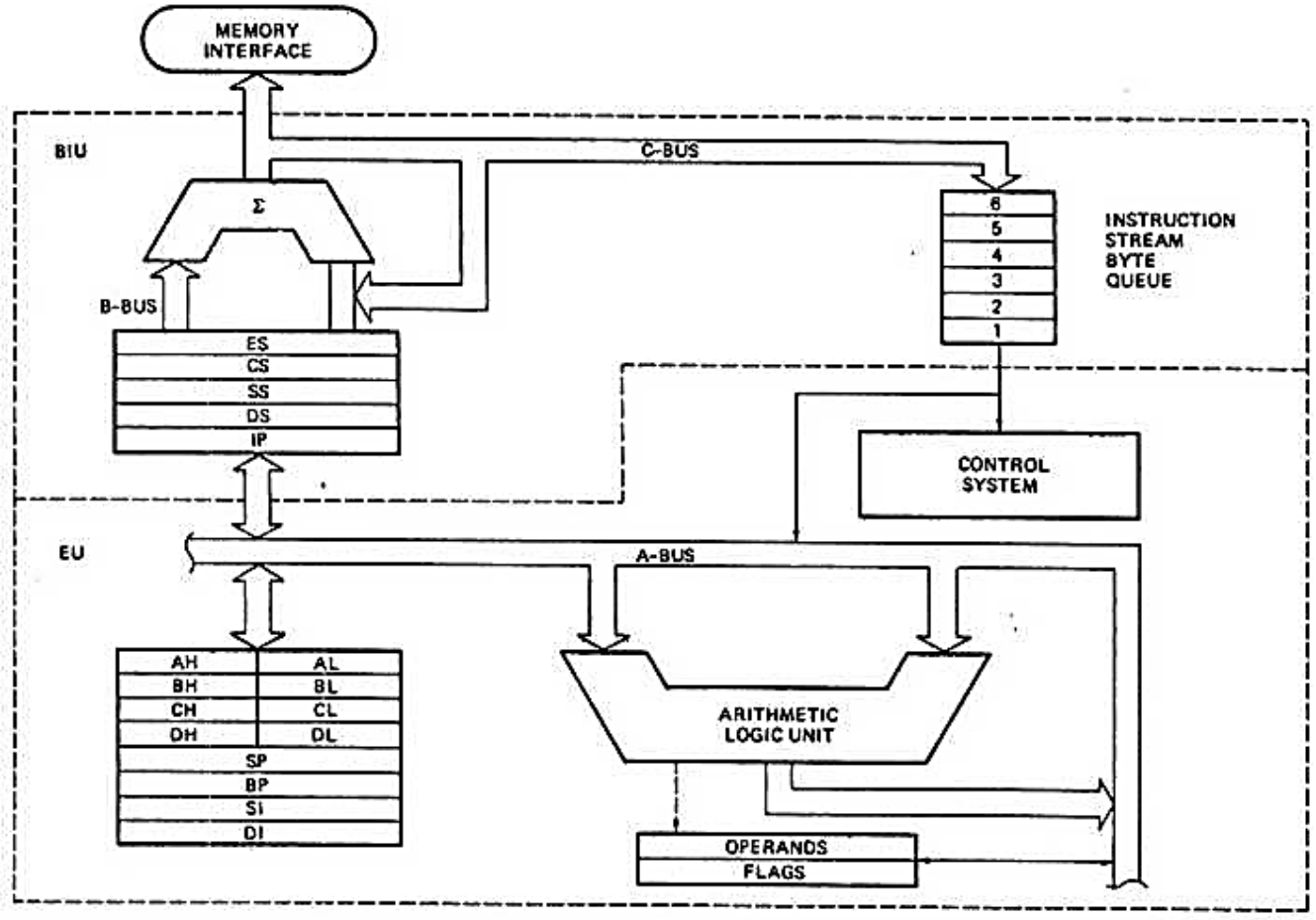
BCD (Binary-Coded Decimal) Data

İkili kodlu onluk(BCD) bilgiler, paketli veya paketsiz formlarda saklanır. Paketlenmiş BCD verileri bayt başına iki basamak olarak ve paketlenmemiş BCD verileri bayt başına bir basamak olarak saklanır. Bir BCD rakamının aralığı 00002 ila 10012 veya 0-9 ondalık arasındadır. Ambalajsız BCD verileri tuş takımından veya klavyeden döndürülür. Paketlenmiş BCD verileri, mikroişlemcinin talimat setinde BCD toplama ve çıkarma için dahil edilen bazı talimatlar için kullanılır.

MİKROİŞLEMCİ MİMARİSİ

8086 İŞLEMCİLERİN ÖNEMLİ ÖZELLİKLERİ

1. **Address Bus (Adres Veriyolu):** x86 işlemciler 20 bit adresleme yoluna sahiptir. Bu da 2 ^{20} byte hafıza yani 1 MB demektir. Adres aralığı **00000H … FFFFFH** değişir.
2. **Data Bus (veri yolu**): 16 bit veri yoluna sahiptir. Aynı zamanda **ALU** ve dahili registerlar 16 bit veri yoluna sahiptir. Bundan **dolayı 8086 işlemciler 16 bit işlemci olarak nitelendirilir**.
3. **Control Bus :** Kontrol veriyolu RD, WR vb. Gibi çeşitli işlemlerin gerçekleştirilmesinden sorumlu sinyalleri taşır.
4. Bellek banklarına sahiptirler. 1 MB'lık tüm bellek, 1 döngüde 16 bit aktarmak için her biri 512 KB'lık 2 banka bölünmüştür. Bankalara Aşağı Banka- Lower Bank (çift) ve Yüksek Banka -High Bank(tek) denir. Örnek AX=AH+AL ,AX=1234h ise AH=12h ve AL=34h
5. Hafıza segmenti 4 segmentten oluşur: Code, Stack, Data ve Extra segment olmak üzere.
6. 256 adet interrupt'a sahiptir.
7. 16 bit IO adresine sahiptir. Bu da yaklaşık 64k adet port demektir.



BIU (Bus Interface Unit), Address Generation Circuit (Adres Üretme Devresi) sahiptir.

Segment adres virtual adres (VA) ,PA vey FA fiziksel adres olmak üzere;

FA=segment adres x 10h +offset adres

Örnek code segement adresi **1234h** ve offset adresi **0005h** olsun. Fiziksel adresi hesaplayalım

FA=CSx10h+offset adres

FA=1234x10+0005

FA=12345h olmuş olur.

Kaydediciler

Programlama modeli 8, 16 ve 32 ve 64 bit kaydedici (yazmaç=register) içerir.

8086 işlemcilerine ait kaydediciler üç grupta toplanabilir:

a. Genel Amaçlı Kaydediciler
b. İşaretçi ve İndis Kaydediciler
c. Segment Kaydediciler

Bu gruba ek olarak, MİB' ye ait olan ve çeşitli durumları gösteren (aritmetik ya da mantıksal işlem sonucu gibi) bir bayrak(flags) kaydedicisi de bulunmaktadır.

a. Genel Amaçlı Kaydediciler: *****

Bu grupta yer alan kaydediciler, programcı tarafından değişik amaçlarla kullanılabilirler. Bunlardan her biri 16-bit ya da 8-bit olarak kullanılabilirler. Bu kaydedicilerin temel fonksiyonları aşağıda anlatılmaktadır:

16 bit	Yüksek Bank (tek)- 8 bit	Düşük bank (çift)- 8 bit
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

AX (Accumulator - Toplam): Çarpma ve bölme işlemleri sırasında işlenenleri ve sonuçları tutar. ALU'daki en önemli yazmaçtır. Aritmetik işlemler bu yazmaç üzerinden yapılır ve sonuç yine burada saklanır. String operasyonlarında akümülatör olarak kullanılır. Tüm IO data In veya OUT transferlerinde kullanılır.

BX (Base - Taban): Hafızada yer alan bir verinin taban (ofset) adresini veya XLAT (translate) komutu ile erişilen bir tablo verisinin taban adresini içermede sık olarak kullanılır. Dolaylı Adresleme modlarında bellek adresini (ofset adresi) tutar.

CX (Count - Sayma): Bir kaydırma (shift) veya döndürme (rotate) gibi işlemlerde, bit sayısını tutmada; string veya LOOP komutundaki işlem sayısını belirtmede döngü sayacı olarak kullanılır. Loop,Rotate,Shift ve String operasyonlarının sayısını tutar.

DX (Data - Veri): Özellikle çarpma işlemlerinden sonra, sonucun yüksek değerli kısmını, bir bölme işleminden önce bölünen sayının yüksek değerli kısmını ve değişken I/O komutunda I/O port numarasını tutma işlemlerinde kullanılır. Çarpma ve Bölme sırasında AX ile 32 bit değerleri tutmak için kullanılır. IO Portunun adresini dolaylı IO adresleme modunda tutmak için kullanılır

b. İşaretçi ve İndis Kaydediciler

Bu kaydediciler genel amaçlı olarak kullanılabilmelerine rağmen,genellikle, hafızada yer alan operand'lara erişimde indis veya işaretçi olarak kullanılırlar.

SP (Stack Pointer – Yığın İşaretçisi): Bir veri yığınının denetiminde kullanılan ve bir sonraki adımda erişilecek olan yığın öğesinin yerini işaret eden yazmaçtır.
SP, Yığının üst kısmının ofset adresini tutar. Yığın, LIFO biçiminde çalışan bir bellek konumlar kümesidir. Yığın, Yığın Segmentindeki hafızada bulunur.Yığın Segmentinin fiziksel adresini hesaplamak için SS register ile SP kullanılır. PUSH, POP, CALL, RET vb. gibi talimatlar sırasında kullanılır. `SS stacksegmentin adresini`
PUSH komutu sırasında **SP 2 azalır**, POP sırasında ise **SP 2 arttırılır**.

BP (Base Pointer – Taban İşaretçisi): Hafızada yer alan bir veri dizisini adreslemede kullanılır.
BP, yığın segmentindeki herhangi bir konumun ofset adresini tutabilir.
Yığının rastgele konumlarına erişmek için kullanılır

SI (Source Index – Kaynak İndisi): String komutlarında kaynak veriyi dolaylı adresleme de kullanılır. Normalde Veri segmenti için ofset adresini tutmak için kullanılır, ancak Segment Geçersiz Kılmayı kullanan diğer segmentler için de kullanılabilir.Dize İşlemleri sırasında veri segmentindeki kaynak verilerin ofset adresini tutar.

DI (Destination Index – Hedef İndisi): String komutlarında hedef veriyi dolaylı adresleme de kullanılır. Normalde Ekstra segment için ofset adresini tutmak için kullanılır, ancak Segment Geçersiz Kılmayı kullanan diğer segmentler için de kullanılabilir. Dize İşlemleri sırasında Ekstra Segmentteki hedefin ofset adresini tutar.

IP (Instruction Pointer – Komut İşaretçisi): Her zaman mikroişlemci tarafından yürütülecek bir sonraki komutu adresleme de kullanılır.

c. Segment Kaydediciler

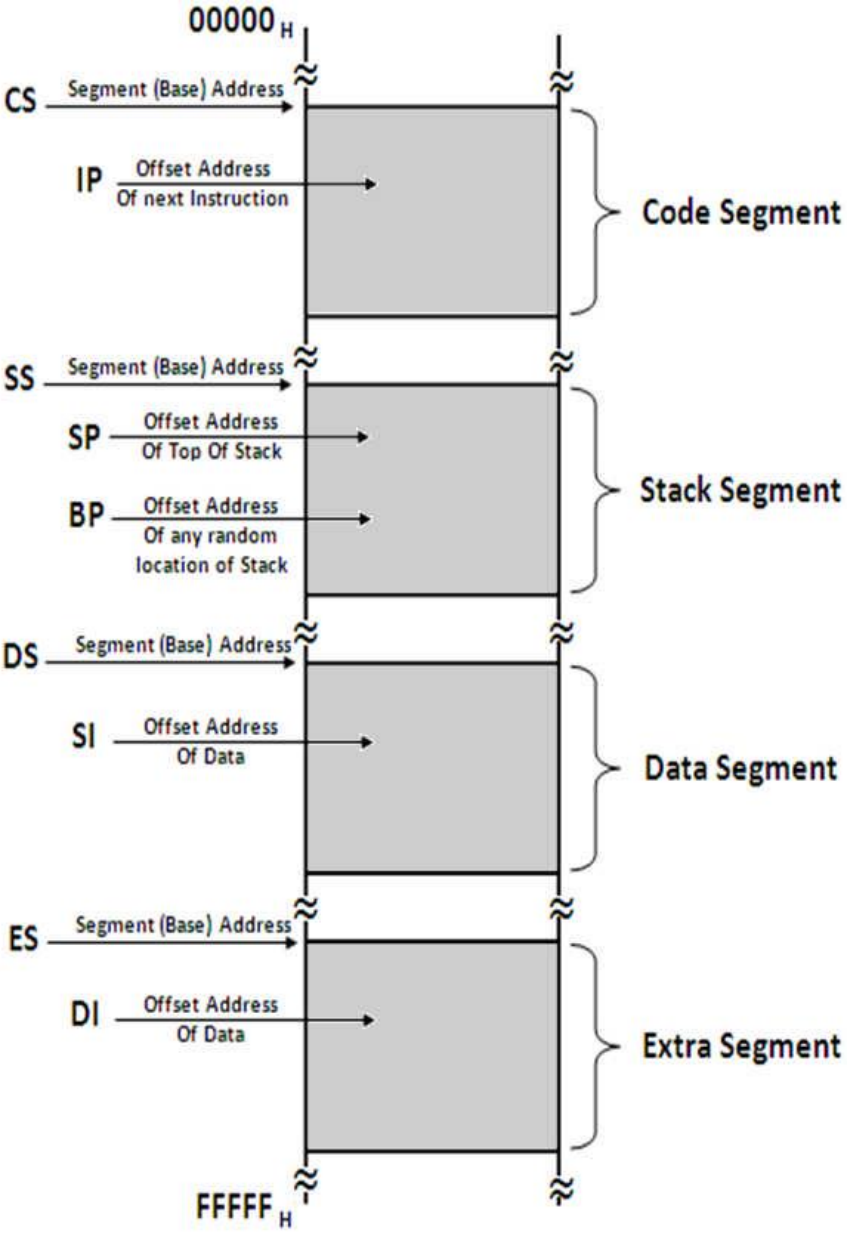
Mikroişlemcideki diğer kaydedicilerle birlikte hafıza adresleri üretmede kullanılırlar. Aşağıda kısaca bu kaydedicilerin görevleri anlatılmaktadır:

CS: Hafızanın, programları ve alt programları tutan bir bölümüdür. CS, program kodunun başlangıç taban adresini belirler. Yani code segmentin başlangıç adresini tutar.

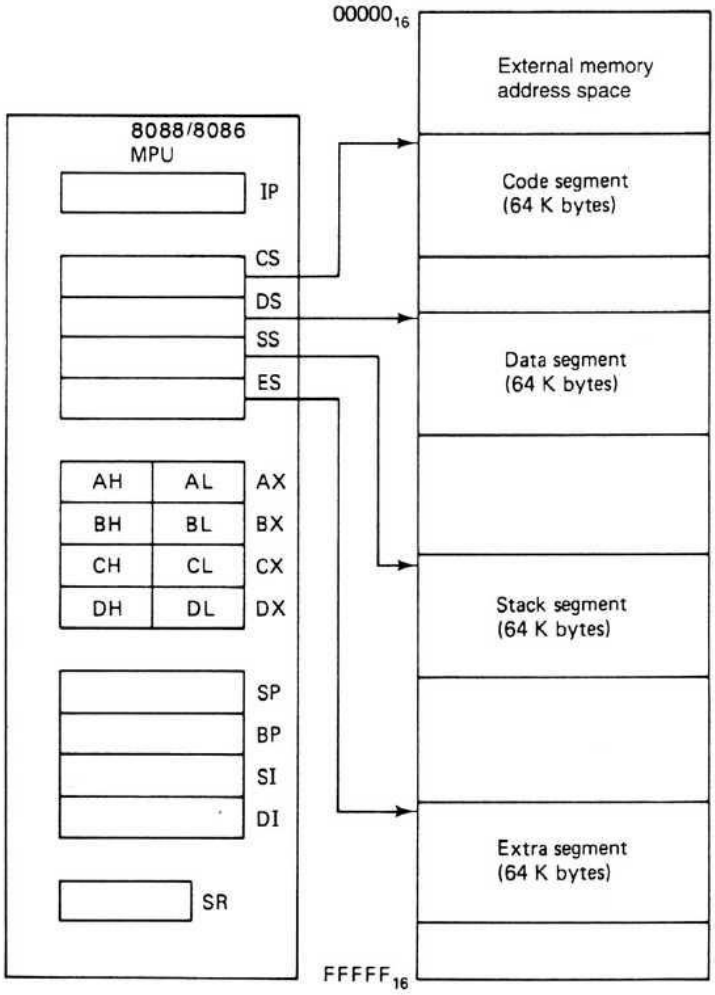
DS : Bir program tarafından kullanılan verilerin bulunduğu adresi hafıza alanıdır. Yani data segmentin başlangıç adresini tutar.

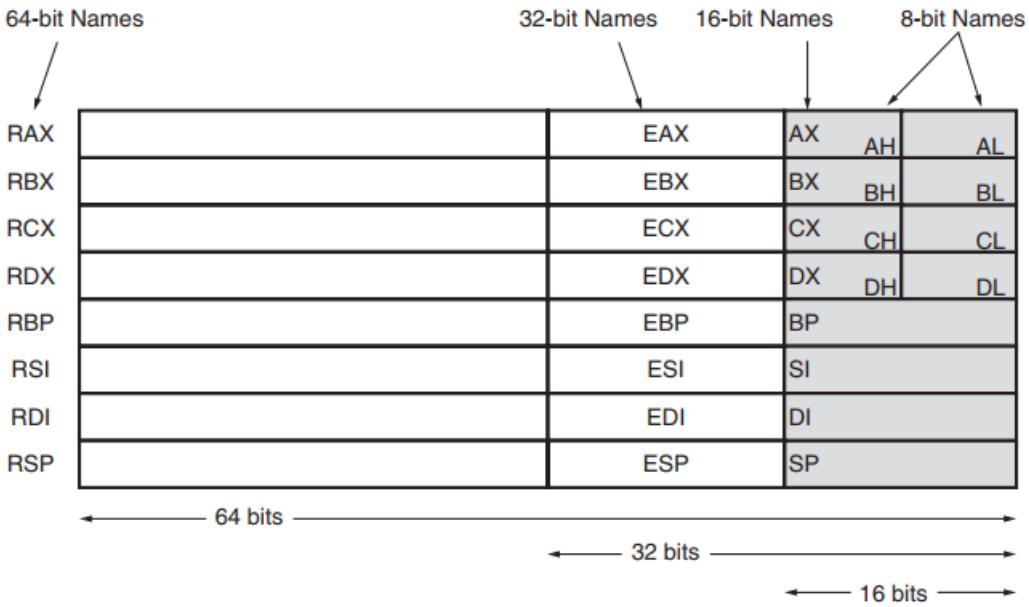
ES : Bazı string komutlarında kullanılan ek veri alanıdır.

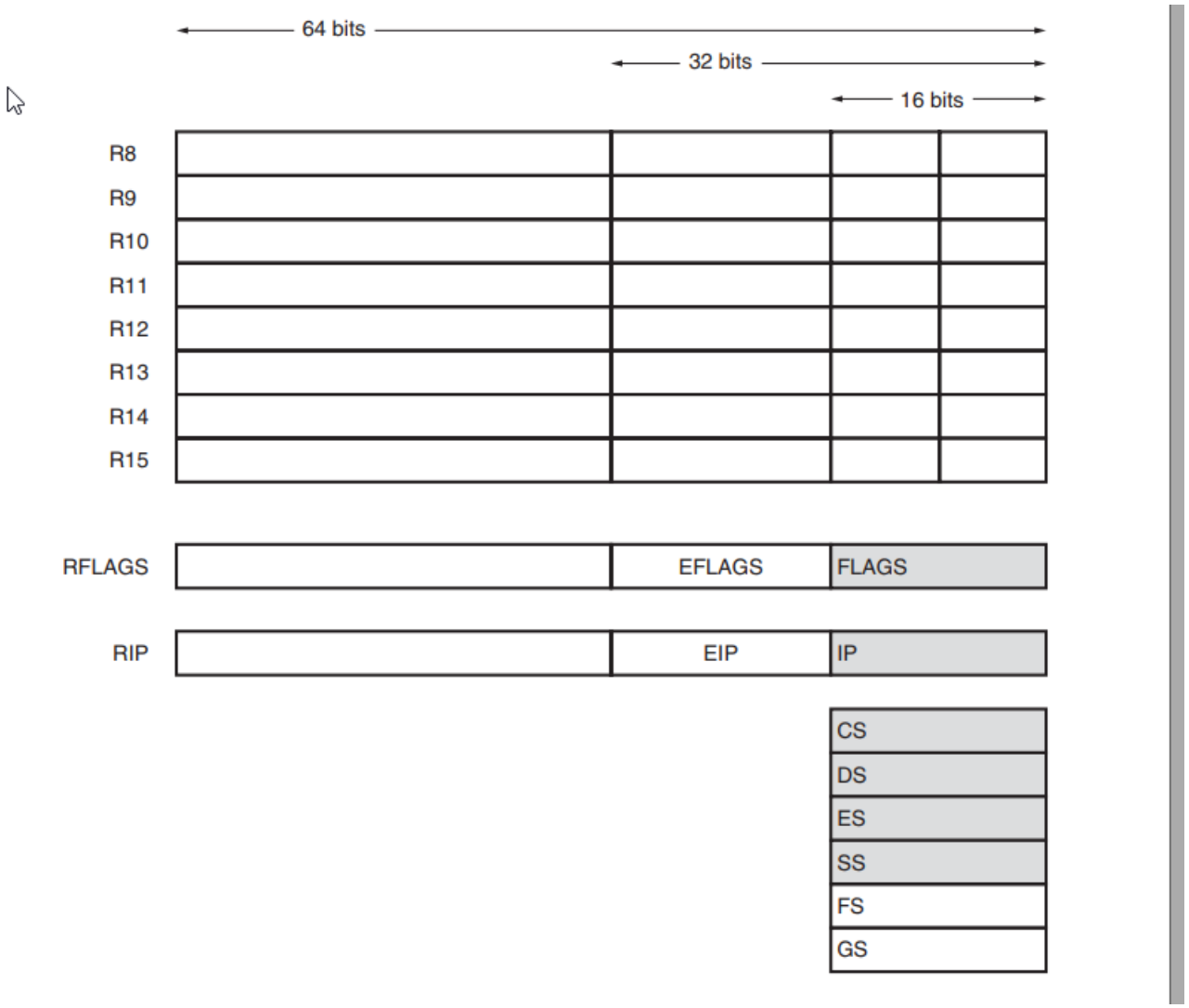
SS :Yığın için kullanılan hafıza alanını belirler. Yığın segmentine yazılacak veya okunacak verinin adresi, SP tarafından belirlenir. BP de SS'de bulunan veriyi adreslemede kullanılır.


00000 H
CS
Segment (Base) Address
IP
Offset Address
Of next Instruction
Code Segment
SS
Segment (Base) Address
SP
Offset Address
Of Top Of Stack
BP
Offset Address
Of any random
location of Stack
Stack Segment
DS
Segment (Base) Address
SI
Offset Address
Of Data
Data Segment
ES
Segment (Base) Address
DI
Offset Address
Of Data
Extra Segment
FFFFF H

8086 SOFTWARE MODEL







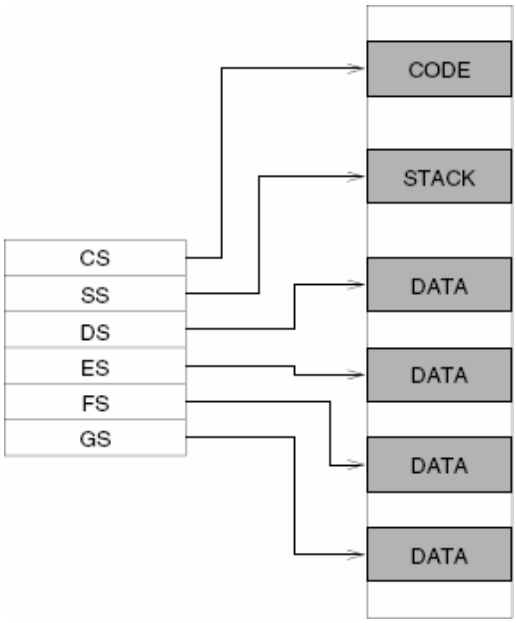
Register	Accumulator	Counter	Data	Base	Stack Pointer	Stack Base Pointer	Source	Destination
64-bit	RAX	RCX	RDX	RBX	RSP	RBP	RSI	RDI
32-bit	EAX	ECX	EDX	EBX	ESP	EBP	ESI	EDI
16-bit	AX	CX	DX	BX	SP	BP	SI	DI
8-bit	AH AL	CH CL	DH DL	BH BL	SPL	BPL	SIL	DIL

1. **Code segment register (CS):** Çalıştırılabilir programın saklandığı belleğin kod segmentindeki bellek konumunu adreslemek için kullanılır.

2. **Data segment register (DS):** Verilerin saklandığı belleğin veri segmentini gösterir, işaret eder.

3. **Extra Segment Register (ES):** Bellekteki başka bir veri segmenti olan bir segmenti de ifade eder.

4. **Stack Segment Register (SS):** belleğin yığın segmentini adreslemek için kullanılır. Yığın bölümü, yığın verilerini depolamak için kullanılan bellek bölümüdür.



Not: Aslında FS ve GS ek segment kaydedicileri aslında ES kopyasıdır. 386 ve sonraki x86 modellerinde yer almaktadır. ES,Fs ve GS hem data hem de code için kullanılabilir.

8 bitlik kaydediciler (registers) AH, AL, BH, BL, CH, CL, DH ve DL'dir. Ve bu iki harfli gösterimler kullanılarak bir komut oluşturulduğunda belirtilir.

Örneğin, bir **ADD AL, AH** komutu AH'nin 8-bit içeriğini AL'ye ekler. (Bu talimat nedeniyle yalnızca AL değişir.).

16 bitlik kayıtlar AX, BX, CX, DX, SP, BP, DI, SI, IP, FLAGS, CS, DS, ES, SS, FS ve GS'dir. İlk 4 16 bitlik register bir çift 8 bitlik kayıt içerdiğini unutmayın. AH ve AL içeren AX buna bir örnektir. Yani aslında AX=AH+AL şeklinde özetlenebilir. 16 bitlik regsiter AX gibi iki harfli adlandırmalarla başvurulur. Örneğin, bir **ADD DX, CX** komutu, CX'in 16 bit içeriğini DX'e ekler. (Bu talimat nedeniyle yalnızca DX değişir.)

Genişletilmiş 32 bit kaydediciler EAX, EBX, ECX, EDX, ESP, EBP, EDI, ESI, EIP ve EFLAGS'tır.
Bu 32 bit genişletilmiş kaydediciler ve 16 bit olan FS ve GS kaydediciler yalnızca 80386 ve sonraki sürümlerde kullanılabilmektedir.

16-bit kaydedicilere, iki yeni 16-bit kayıt için FS veya GS adlarıyla ve 32-bit kayıtlar için üç harfli bir ad verilir.

Örneğin, bir **ADD ECX, EBX** komutu EBX'in 32 bit içeriğini ECX'e ekler.(Bu talimat nedeniyle yalnızca ECX değişir.)

Bazı kaydediciler genel amaçlı veya çok amaçlı kayıtlar iken, bazılarının özel amaçları vardır.Çok amaçlı kaydediciler arasında EAX, EBX, ECX, EDX, EBP, EDI ve ESI sayılabilir.

Bu kaydediciler çeşitli veri boyutlarına sahiptir (bayt, kelimeler veya çift sözcükler) ve bir program tarafından dikte edildiği gibi hemen hemen her amaç için kullanılır.

Register Size	Override	Bits Accessed	Example
8 bits	B	7–0	MOV R9B, R10B
16 bits	W	15–0	MOV R10W, AX
32 bits	D	31–0	MOV R14D, R15D
64 bits	—	63–0	MOV R13, R12

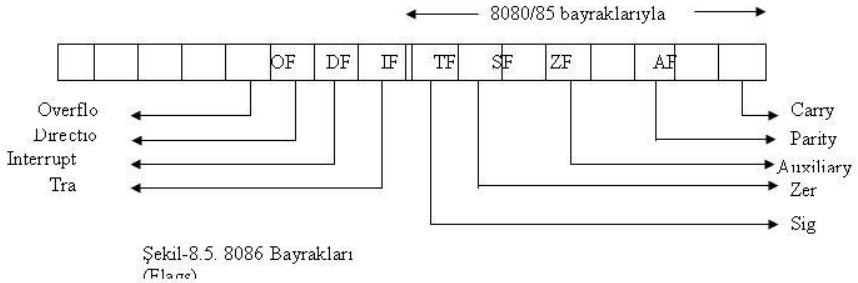
BAYRAKLAR (FLAGS)

Flag Register her biri 16 bittir. 8086 işlemcilerde 9 adet bayrak vardır. İki tip bayrak vardır: **Status (condition)** 6 adet ve **Control** 3 adettir.

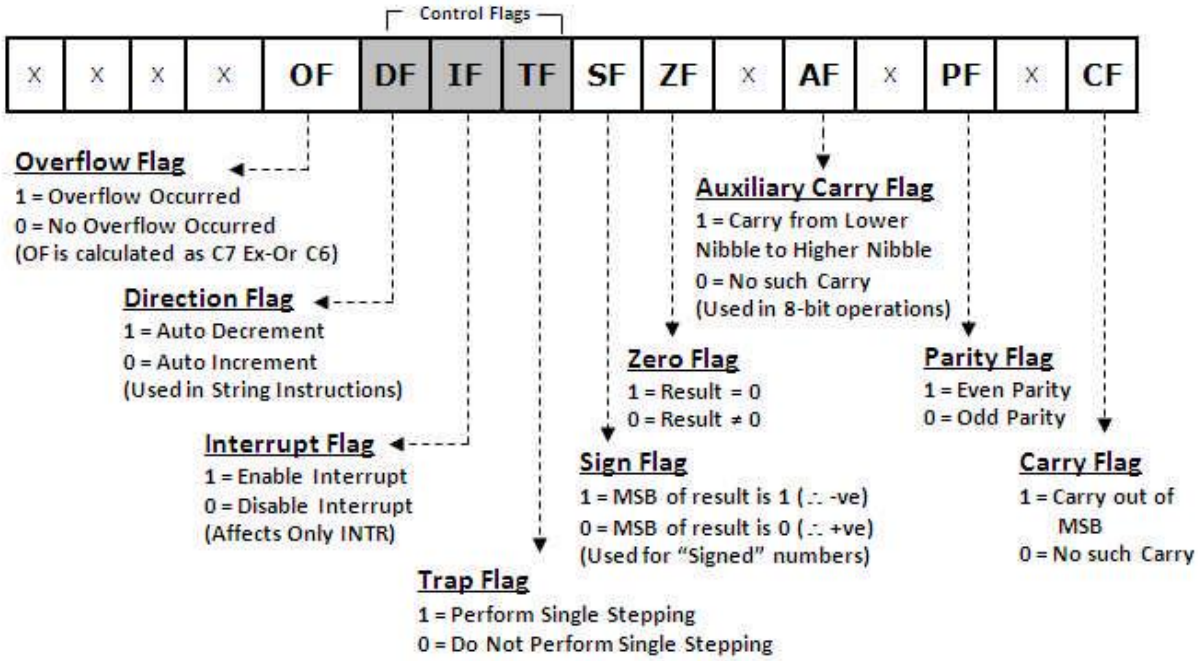
Status (durum) bayrakları her aritmetik veya mantıksal işlemden sonra yani ALU'dan etkilenir. Mevcut sonucun durumunu verir.

Control (kontrol) bayrakları ise belirli işlemleri kontrol etmek için kullanılır.Programcı tarafından değiştirilirler.

Bayraklar, işlemcinin çalışmasını belirler ve çalışması sırasındaki durumunu yansıtır. Aşağıdaki resimde 8086 işlemcisinin bayraklar saklayıcısını göstermektedir. Bu bayrakların düşük 8-bitlik kısmı 8085 işlemcisindekiyle özdeştir. Yeni olan dört bayraktan üç tanesi gerçek kontrol bayraklarıdır.



Şekil-8.5. 8086 Bayrakları



A. DURUM BAYRAKLARI (STATUS FLAGS)

1) Carry: Bir aritmetik işlemde, toplamadan sonraki eldeyi veya çıkarmadan sonraki ödüncü belirtir. Programlarda hata durumu, özel işlem durumları ve sonuçlarıyla ilgili boolean bayrak olarak da kullanılır. D7 biti 8 bit,D15 biti de 16 bit operasyonlarında elde bayrağı olarak kullanılır.

2) Parity: Tek eşlik işlemi, lojik 0;çift eşlik işlemi lojik 1 ile gösterilir. Eşlik, çift veya tek olarak belirtilen bir byte veya word'teki birlerin sayısıdır.

3) Auxiliary carry: Yapılan bir işlem sonucunda, bit pozisyonları 3 ve 4 arasında olan (en sağdaki bit sıfır pozisyonundadır) toplamadan sonraki eldeyi veya çıkarmadan sonraki ödüncü belirtir. Düşük dört bit bir taşıma üretilirse ayarlanır.
Yalnızca DAA ve DAS gibi 8 bitlik işlemlerde kullanılır.

4) Zero: Bir aritmetik ve mantıksal işlem sonucunun sıfır olduğunu belirtir. Eğer Z=1 ise sonuç sıfırdır; Z=0 ise sonuç sıfır değildir.

5) Sign: Bir toplama veya çıkarma işleminden sonra, sonucun aritmetik işaretini belirtir. Eğer S=1 ise işaret 1'lenir veya negatiftir. Eğer S=0 ise; işaret temizlenir veya pozitiftir. Bayrakları etkileyen bir komuttan sonra, en değerli bit(MSB) pozisyonu S bit''ne yerleştirilir.

6) **Overflow**: Taşma, işaretli sayıların toplandığında veya çıkartıldıklarında oluşan bir durumdur. Taşma, işlem sonucunun hedef kaydediciye sığmadığını gösterir. İşaretli bir işlemin sonucunun, onu temsil etmek için mevcut bit sayısına sığmayacak kadar büyük olması durumunda ayarlanır. INTO (Interrupt on Overflow - Taşmada Kesinti) talimatı kullanılarak kontrol edilebilir.

B. KONTROL BAYRAKLARI (CONTROL FLAGS)

a. Trap: İzleme modunu ayarlar. Mikroişlemci her komuttan sonra bir interrupt alır . Böylece program hata ayıklanabilir (**debug**). Eğer Trap bayrağı 1'lenmiş ise, tüm devre hata takip işlemi devreye girer.

b. Interrupt: Mikroişlemci tüm devresinin kesme isteği giriş bacağı INTR, harici kesme isteği işlemini kontrol eder. INTR kesmesini maskelemek (devre dışı bırakmak) veya maskesini kaldırmak (etkinleştirmek) için kullanılır.

c. Direction: String komutları yürütülürken DI ve/veya SI kaydedicilerinin artırılması veya azaltılması işlemlerinin seçimini kontrol eder. Bu bayrak ayarlanırsa, String İşlemlerinde SI ve DI otomatik azaltma modundadır.

ADRESLEME MODLARI

Mikroişlemci için verimli yazılım geliştirme, her bir talimat tarafından kullanılan adresleme modlarına tam bir aşinalık gerektirir.

```
mov : değeri kopyalama
```

Bu bölümde, veri adresleme modlarını tanımlamak için MOV (veri taşıma) komutu kullanılır.

MOV komutu, baytları veya veri kelimelerini 8086 ile 80286 arasındaki iki kayıt arasında veya kayıtlar ile bellek arasında aktarı yapar.

Baytlar, kelimeler (Word=2 byte) veya çift kelimeler (dword) 80386 ve üzeri sürümlerde bir MOV tarafından aktarılır.

Program bellek adresleme modlarını açıklarken, CALL ve JUMP talimatları program akışının nasıl değiştirileceğini gösterir.

Veri adresleme modları, 8086'dan 80286'a kadar olan mikroişlemcilerinde, anında (immediate), doğrudan (direct), kayıt dolaylı (register indirect), baseplus indeksi (baseplus index), kayıt bağıl (register relative) ve baz bağıl artı-dizinini (base relative plus-index) içerir.

80386 ve üstü, bellek verilerinin adreslenmesi için ölçeklenmiş bir dizin modu (scaled-index mode) da içerir.

Program hafızası adresleme modları, programın göreceli (reative), doğrudan (direct) ve dolaylı (indirect) şeklindedir.

Bu bölümde, PUSH ve POP talimatları ve diğer yığın işlemlerinin anlaşılması için yığın belleğin çalışması açıklanmaktadır.

DATA ADRESLEME MODLARI (DATA ADDRESSING MODE)

MOV komutu çok yaygın ve esnek bir komut olduğundan, veri adresleme modlarının açıklanması için bir temel sağlar.

Aşağıdaki şekil MOV komutu ve veri akışının yönünü tanımlar.

Kaynak sağda ve hedef solda olmak üzer , MOV opcode'unun yanında. (Bir opcode veya işlem kodu mikroişlemciye hangi işlemin gerçekleştirileceğini söyler.)

Bir virgülün hedefi her zaman bir komutta kaynaktan ayırdığına dikkat edin. Ayrıca, MOVS komutu dışında herhangi bir komutla bellekten belleğe aktarımlara izin verilmediğini unutmayın.

```
00 ADD
01 SUB
10 MUL
11 DIV
```

```
MOV 35
ADD 48
```

Not: **"opcode"** (operation code — makine dili komutunun gerçekleştirilecek işlemi belirtilen kısmı) denir. Tüm talimatlara uygulanan bu akış yönü ilk başta gariptir. 35 sayısının işlemciye veriyi bir memory hücresinden diğerine taşıması ile ilgili talimat olduğunu söylediğimiz örneği hatırlayın. Assembly bu talimatı "Move"'un kısaltması şeklinde "MOV" komutuna atar. 48 yani bir sayının diğerine eklenmesi talimatını "ADD" komutu olarak, 12 ile belirtilen "OR" mantıksal operasyonunu"ORL" komutu olarak isimlendirir.
Programcı komutlarını yazdıktan sonra **"Assembler"** olarak adlandırılan aracı çalıştırır. Bu araç sembolleri (MOV, ADD vs) işlemcinin anlayıp çalıştırabileceği sayısal kodlara (opcode) çevirir.
Doğal olarak, şeylerin soldan sağa hareket ettiğini varsayıyoruz, oysa burada sağdan sola hareket ediyorlar.

BÖLÜM TEKRARI

1) Segmentasyon, hafızayı, segment adı verilen mantıksal olarak farklı parçalara bölmek anlamına gelir.

2) 8086, 20 bit adres veriyoluna sahiptir, bu nedenle 220 Byte yani 1MB belleğe erişebilir.

3) Ancak bu aynı zamanda Fiziksel adresin artık 20 bit olacağı anlamına gelir.

4) Bayt uyumlu bir sayı olmadığı için 20 bit adresle çalışmak mümkün değildir. (20 bit iki buçuk bayttır).

5) Bu uyumsuz sayıyla çalışmayı önlemek için, belleğin sanal bir modelini oluşturuyoruz.

6) Burada bellek 4 bölüme ayrılmıştır: Kod, Yığın Verileri ve Ekstra.

7) Bir segmentin maksimum boyutu 64 KB ve minimum boyut 16 bayttır.

8) Artık programcı her konuma bir SANAL ADRES ile erişebiliyor.

9) Sanal Adres, Segment Adresi ve Ofset Adresinin bir kombinasyonudur.

10) Segment Adresi, segmentin bellekte nerede bulunduğunu gösterir (temel adres)

11) Ofset Adresi, segment içindeki hedef konumun ofsetini verir.

12) Her ikisi de, Segment Adresi ve Ofset Adresi her biri 16 bit olduğundan, her ikisi de uyumlu numaralardır ve programcı tarafından kolayca kullanılabilir.

13) Ayrıca, Segment Adresi, segmenti başlatmak için sadece programın başında verilir. Daha sonra sadece ofset adresi veriyoruz.

14) Bu nedenle, 1 MB belleğe programın çoğu kısmı için yalnızca 16 bitlik bir ofset adresi kullanarak erişebiliriz. Segmentasyonun avantajı budur.

15) Ayrıca, Kod, yığın ve Verilerin farklı segmentlere bölünmesi hafızayı daha düzenli hale getirir ve aralarında yanlışlıkla üzerine yazma yapılmasını önler.

16) Bir segmentin Maksimum Boyutu 64 KB'dir, çünkü ofset adresleri 16 bittir. 216 = 64 KB.

17) Bir segmentin maksimum boyutu 64KB olduğundan, programcı 1 MB'nin tamamı kullanılıncaya kadar birden fazla Kod / Yığın / Veri segmenti oluşturabilir, ancak şu anda her türden yalnızca biri aktif olacaktır.

18) Fiziksel adres, aşağıdaki formül kullanılarak mikroişlemci tarafından hesaplanır:

FİZİKSEL ADRES = SEGMENT ADRESİ X 10H + OFSET ADRESİ

19) Örn: Segment Adresi = 1234H ve Ofset Adresi 0005H ise

Fiziksel Adres = 1234H x 10H + 0005H = 12345H

20) Bu formül otomatik olarak bir segmentin minimum boyutunun 10H bayt olmasını sağlar (10H = 16 Bayt)

Kod Segmenti

- Bu bölüm yürütülecek programı tutmak için kullanılır.
- Talimatlar Kod Segmentinden getirilir.
- CS kaydı, bu segment için 16 bit temel adresi içerir.
- IP kaydı (Yönerge İşaretçisi) 16 bitlik ofset adresini tutar.

Veri Segmenti

- Bu segment genel verileri tutmak için kullanılır.
- Bu segment ayrıca dize işlemleri sırasında kaynak işlenenleri de tutar.
- DS kaydı, bu segment için 16 bit temel adresi tutar.
- BX kaydı, bu segment için 16 bitlik sapmayı tutmak için kullanılır.
- SI kaydı (Kaynak Dizini), Dize İşlemleri sırasında 16 bitlik ofset adresini tutar.

Yığın Segmenti

- Bu segment, LIFO tarzında çalışan Yığın belleğini tutar.
- SS, Temel adresini tutar.
- SP (Yığın İşaretçisi), Yığının Üstünün 16 bitlik ofset adresini tutar.
- BP (Baz İşaretçi) Rastgele Erişim sırasında 16 bitlik ofset adresini tutar.

Ekstra Segment

- Bu segment genel verileri tutmak için kullanılır
- Ayrıca, bu segment Dize İşlemleri sırasında hedef olarak kullanılır.
- ES, Temel Adresi tutar.
- DI, dize işlemleri sırasında ofset adresini tutar.

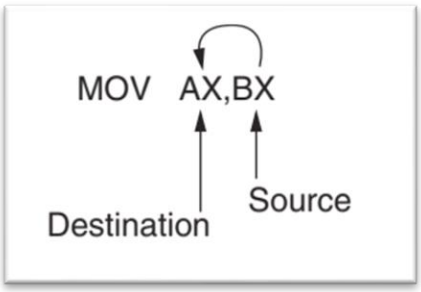
Segmentasyonun Avantajları:

1) Programlayıcının sadece 16 bit adres kullanarak 1MB'ye erişmesine izin verir.
2) Talimatları, Verileri ve Yığını ayrı ayrı saklamak için belleği mantıksal olarak böler.

Segmentasyonun Dezavantajı:
1) Toplam bellek 16 * 64 KB olmasına rağmen, bir seferde sadece 4 * 64 KB belleğe erişilebilir.

Type	Instruction	Source	Address Generation	Destination
Register	MOV AX,BX	Register BX		Register AX
Immediate	MOV CH,3AH	Data 3AH		Register CH
Direct	MOV [1234H],AX	Register AX	DS × 10H + DISP 10000H + 1234H	Memory address 11234H
Register indirect	MOV [BX],CL	Register CL	DS × 10H + BX 10000H + 0300H	Memory address 10300H
Base-plus-index	MOV [BX+SI],BP	Register SP	DS × 10H + BX + SI 10000H + 0300H + 0200H	Memory address 10500H
Register relative	MOV CL,[BX+4]	Memory address 10304H	DS × 10H + BX + 4 10000H + 0300H + 4	Register CL
Base relative-plus-index	MOV ARRAY[BX+SI],DX	Register DX	DS × 10H + ARRAY + BX + SI 10000H + 1000H + 0300H + 0200H	Memory address 11500H
Scaled index	MOV [EBX+2 × ESI],AX	Register AX	DS × 10H + EBX + 2 × ESI 10000H + 00000300H + 00000400H	Memory address 10700H

Notes: EBX = 00000300H, ESI = 00000200H, ARRAY = 1000H, and DS = 1000H



1. KAYDEDİCİ (REGISTER) ADRESLEME

Register adresleme, kaynak bayttan veya bir bellek konumunun içeriğinden bir byte veya word bir kopyasını hedef kayıt kaydedici veya bellek konumuna aktarır.

(Örnek: MOV CX, DX komutu, DX kaydının word boyutundaki içeriğini CX kaydedicisine kopyalar.)

80386 ve üzeri sürümlerde, kaynak register veya hafıza konumundan hedef kaydedici veya hafıza konumuna bir çift word aktarılabilir.

(Örnek: MOV ECX, EDX komutu, EDX kaydının çift word içeriğini ECX kaydına kopyalar.)

64 bit modunda çalıştırılan Pentium 4'te 64 bitlik kayıtlara da izin verilir.

Bir örnek, RCX kaydının dörtlü içeriğinin bir kopyasını RDX kaydına aktaran MOV RDX, RCX komutudur.

2. İVEDİ (IMMEDIATE) ADRESLEME

Anında adresleme, kaynağı, bir anlık byte, word, doubleword veya quadword hedef registerine veya bellek konumuna aktarır.

(Örnek: MOV AL, 22H komutu, byte boyutlu bir 22H'yi AL kaydına kopyalar.)

80386 ve üzeri sürümlerde, bir acil doubleword bir veri register veya bellek konumuna aktarılabilir.

(Örnek: MOV EBX, 12345678H komutu doubleword boyutlu bir 12345678H komutunu 32 bit genişlikli EBX kaydına kopyalar.)

Not : Pentium 4 veya Core2'nin 64 bit işletiminde, yalnızca bir MOV acil talimatı 64 bit doğrusal adres kullanarak bellekteki herhangi bir konuma erişime izin verir.

3. DOĞRUDAN (DIRECT) ADRESLEME

Doğrudan adresleme, bir byte veya word bellek konumu ile register arasında taşır.

Komut kümesi, MOVS komutu dışında hafızadan belleğe aktarımı desteklemez.

(Örnek: MOV CX, LIST komutu, bellek konumu LIST'in word boyutundaki içeriğini CX kaydına kopyalar.)

Not : 80386 ve üzeri sürümlerde çift sözcük boyutlu bir bellek konumu da ele alınabilir. (Örnek: MOV ESI, LIST komutu, 32 bitlik bir sayıyı depolayan ardışık bayt belleğini, LIST konumundan ESI kaydedicisine kaydeder.)64 bit modundaki doğrudan bellek talimatları tam 64 bit doğrusal adres kullanır.

4. REGISTER DOLAYLI ADRESLEME (REGISTER INDIRECT ADRESSING)

Register dolaylı adresleme, bir index veya base register tarafından adreslenen,register ve bir bellek konumu arasında bir byte veya word aktarır. Index ve base register 'lar BP,BX,DI ve SI'dir.

(Örnek: MOV AX, [BX] komutu, word boyutlu verileri BX tarafından endekslenen veri segmenti ofset adresinden AX kaydına kopyalar.)

80386 ve üzeri sürümlerdeki işlemcilerde , bir byte, word veya doubleword herhangi bir register tarafından adreslenen veri ,register ile bir bellek konumu arasında aktarılır: EAX, EBX, ECX, EDX, EBP, EDI veya ESI.

(Örnek: MOV AL, [ECX] komutu AL'yi ECX içeriği tarafından seçilen veri bölümü ofset adresinden yükler.)

5. TABAN-ARTI-INDİS (BASE PLUS INDEX ADDRESSING) ADRESLEME

Base Plus Index Adreslemede,base register (BP veya BX) ve index register (DI veya SI) ile adreslenen byte ya da word veriyi register ve bellek konumu arasında veriyi aktarır.

(Örnek: MOV [BX+DI], CL komutu, CL kaydının bayt boyutlu içeriğini BX artı DI tarafından adreslenen veri segmentinin bellek konumuna kopyalar.)

6. SAKLAYICI GÖRECELİ (REGISTER RELATİVE) ADRESLEME

Bu adreslemede, index veya base plus index yerdeğiştirme tarafından adreslenen byte veya word 'ü register ile bellek konumu arasında taşır (MOV),kopyalar.

Örnek: MOV AX,[BX+4] ;BX+4 adresindeki veriyi data segmentteki AX'e kopyalar.

MOV AX,ARRAY[BX] ; BX 'teki diziyi data segmentteki AX'e yükler.

7. TABAN GÖRECELİ-ARTI-İNDİS (BASE RELATIVE PLUS INDEX) ADRESLEME

Bu adreslemede base ve index register artı tarafıdan adreslenen byte veya word'ü regster ile bellek arasında aktarır.

Örnek : MOV AX, ARRAY[BX+DI]

MOV AX,[BX+DI+4]

SCALED INDEX ADDRESSING