ASSEMBLY KOMUTLARI

Aşağıda Tabloda 8086 mikroişlemcisinin değişik işlemleri ve bunlarda otomatik olarak kullanılan saklayıcılar görülmektedir.

İşlemlere göre değişik segment adres kaynakları.			
İşlem Çeşidi	***Segment***	***Alternatif Segment***	***Ofset***
Komut okuma	CS	yok	IP
Veri İşlemi (aşağıdakiler hariç)	DS	CS,ES veya SS	çeşitli
String kaynak	DS	CS,ES veya SS	SI
String hedef	ES	yok	DI
Yığın işlemi	SS	yok	SP
BP taban saklayıcı olarak kullanıldığında	SS	CS, ES veya SS	çeşitli

Asembly Dili	***Yapılan İşlem***
MOV Alıcı, Verici	**Alıcı = Verici** MOV AX, CX : AX = CX MOV CX, 500h : CX = 0500 MOV AL, [CX] : AL = [CX] Adresindeki değer yüklenir
XCHG opr1, opr2	opr1 ve opr2'nin içeriklerini yer değiştirir.
XLAT	DS:BX adresindeki dizi içerisinden n. elemanı bulur. AL=n konularak komut işletilir. Bulunan sonuç yine AL içerisine konur.
PUSH Gönderici	16 bit Gönderici değeri Stakta saklanır. **SP değeri 2 azalır** PUSH AX : AX değeri Stakta saklanır. [SP]=AX ve SP=SP-2
POP Alıcı	Staktaki 16 bit değer, Alıcı içine aktarılır. **SP değeri 2 artar** POP AX : Staktaki 16 bit değer AX'e aktarılır. AX=[SP] ve SP=SP + 2
IN AL, IPORT	IPORT'tan 8-bit veri AL'ye okunur
IN AX, IPORT	IPORT'tan 16-bit veri AX'ye okunur
OUT OPORT, AL	AL'nin içeriği OPORT'a gönderilir
OUT OPORT, AX	AX'in içeriği OPORT'a gönderilir
LEA **AX**, *SUBADR*	**AX**, *SUBADR'* in **Ofset** adresiyle yüklenir.
LDS **DI**, *LIST*	**DI**, *LIST*'in **Ofset** adresiyle yüklenir ve **DS**, *LIST*'in **Segment** adres ile yüklenir.
LES **BX**, *VEC1*	**BX**, *VEC1* 'in **Ofset** adresiyle yüklenir ve **ES**, *VEC1*'in **Segment** adres ile yüklenir.
LAHF	Flag(Bayrak) saklayıcısının düşük 8-bitini AH saklayıcısına yükler
SAHF	AH saklayıcısını, Flag(Bayrak) saklayıcısının düşük 8-bitine yükler
PUSHF	Bayrak saklayıcısının değerini Stağa saklar. **SP değeri 2 azalır.** PUSHF : [SP]= Flag ve SP=SP-2
POPF	Staktaki 16 bit değer, Flag(Bayrak) saklayıcısına alınır. **SP değeri 2 artar.** POPF : Flag=[SP] ve SP=SP + 2

DÖRT İŞLEM KOMUTLARI	
ADD Alıcı, Verici	Alıcı = Alıcı + Verici ADD AL,CL : AL = AL + CL ADD BL,20h : BL = BL + 20h ADD [BX],CL : [BX] = [BX] + CL ; [BX] bu adresteki değer
ADC Alıcı, Verici	Alıcı = Alıcı + Verici + [Carry] ADC AX,BX : AX = AX + BX + [Carry]
SUB Alıcı, Verici	Alıcı = Alıcı - Verici SUB AL,CL : AL = AL – CL SUB CX,400h : CX = CX – 0400h
SBB Alıcı, Verici	Alıcı = Alıcı – Verici - [Carry] SBB AL,CL : AL = AL – CL- [Carry]
MUL DL	AL'de bulunan işaretsiz tamsayı DL ile çarpılır. Çarpım AX'te bulunur. **AX** = **AL** * DL
MUL BX	AX'de bulunan işaretsiz tamsayı BX ile çarpılır. Çarpım DX:AX'te bulunur. **DX:AX** = **AX** * BX
IMUL DL	AL'de bulunan işaretli tamsayı DL ile çarpılır. Çarpım AX'te bulunur. **AX** = **AL** * DL
IMUL BX	AX'de bulunan işaretli tamsayı BX ile çarpılır. Çarpım DX:AX'te bulunur. **DX:AX** = **AX** * BX
DIV DL	**AX**'de bulunan işaretsiz tamsayı DL ile bölünür. Bölüm *Sonucu* **AL**'ye, *Kalan* **AH**'a konur.
DIV CX	**DX:AX**'de bulunan işaretsiz tamsayı CX ile bölünür. Bölüm Sonucu **AX'**e, Kalan **DX**'e konur.
IDIV DL	**AX**'de bulunan işaretli tamsayı DL ile bölünür. Bölüm Sonucu **AL**'ye, Kalan **AH**'a konur.
IDIV CX	**DX:AX**'de bulunan işaretli tamsayı CX ile bölünür. Bölüm Sonucu **AX**'e, Kalan **DX**'e konur.
DECİMAL DÜZENLEYİCİ KOMUTLARI	
AAA	İki desimal sayının toplamasından sonra sonuç üzerinde (AL) düzenleme yaparak, sonucun paketlenmemiş desimal sayılar olmasını (AH ve AL' ye) sağlar.
DAA	İki desimal sayının toplamasından sonra sonuç üzerinde (AL'deki) düzenleme yaparak, sonucun paketlenmiş desimal sayılar olmasını (AL' ye) sağlar.
AAS	İki desimal sayının çıkarılmasından sonra sonuç üzerinde (AL) düzenleme yaparak, sonucun paketlenmemiş desimal sayılar olmasını (AH ve AL' ye) sağlar.
DAS	İki desimal sayının çıkarılmasından sonra sonuç üzerinde (AL'deki) düzenleme yaparak, sonucun paketlenmiş desimal sayılar olmasını (AL' ye) sağlar.
AAM	İki desimal sayının çarpımından sonra sonuç üzerinde (AL) düzenleme yaparak, sonucun paketlenmemiş desimal sayılar olmasını (AH ve AL' ye) sağlar.

AAD	Bölme işleminden önce, AX teki paketlenmemiş desimal sayıyı, AL ye paketlenmiş hex olarak atar.
EK ARİTMETİKSEL KOMUTLARI	
CMP opr1, opr2	Mantıksal karşılaştırma komutu. Bundan sonra Jump komutları kullanılarak mantıksal sorgu yapılır. CMP AL, BL JL git ; AL<BL ise, ***git*** etiket ismine gidilir.
INC opr	Bir artırma komutu. INC BX ; BX=BX + 1
DEC opr	Bir eksiltme komutu. DEC BL ; BL=BL - 1
NEC opr	Negatifini alma komutu. NEC DL ; DL = - DL
CBW	AL deki değeri (1 Byte), AX' e (2 Byte=1 Word) genişletir. 1 bayt sayıyı 2 bayt yapar.
CWD	AX deki değeri (1 Word), DX:AX' e (2 Word =1 Double Word) genişletir.
LOJİK İŞLEM KOMUTLARI	
NOT opr	Mantıksal tersini alır. (0'lar 1, 1'ler 0 yapılır.) MOV AL, 2A ; AL=2A NOT AL ; AL=D5
OR opr1, opr2	Mantıksal VEYA işlemi yapar. OR AL, BH ; AL = (AL) OR (BH)
AND opr1, opr2	Mantıksal VE işlemi yapar. AND AL, BH ; AL = (AL) AND (BH)
XOR opr1, opr2	Mantıksal XOR işlemi yapar. XOR AL, BH ; AL = (AL) XOR (BH)
SHIFT(KAYDIRMA) KOMUTLARI	
SHR	**SHR Opr, CL** ; CL'deki sayı kadar Opr değerini sağa kaydırır, sol taraftan sıfır ekler.
SHL	**SHL Opr, CL** ; CL'deki sayı kadar Opr değerini sola kaydırır, sağ taraftan sıfır ekler.
ROTATE(DÖNDÜRME) KOMUTLARI	
ROL	**ROL Opr, CL** ; CL'deki sayı kadar Opr değerini sola döndürür.
ROR	**ROR Opr, CL** ; CL'deki sayı kadar Opr değerini sağa döndürür.
RCL	**RCL Opr, CL** ; CL'deki sayı kadar Opr değerini Carry ile beraber sola döndürür.
RCR	**RCR Opr, CL** ; CL'deki sayı kadar Opr değerini Carry ile beraber sağa döndürür.

DALLANMA KOMUTLARI	
Jump komutları bir yere yönlendirme işlemi yapar. Genel yazılışları; *JUMP etiket_ismi* olarak yazılırlar. Kendilerinden önceki komutların sonucuna göre işlem yaparlar.	
JMP	Şartsız gitme (dallanma, yönlendirme) komutu. JMP atla ; şartsız olarak ***atla*** etiket ismine gider.
JE	Kendisinden önceki komutun sonucu ***eşit*** ise gider. CMP AL, BL JE atla ; AL=BL ise ***atla*** etiket ismine gider.
JZ	Kendisinden önceki komutun sonucu ***sıfır*** ise gider. ADD AL, BL JZ atla ; AL=0 ise ***atla*** etiket ismine gider
JL	Kendisinden önceki komutun sonucu ***küçük*** ise gider. CMP AL, BL JL atla ; AL<BL ise ***atla*** etiket ismine gider.
JLE	Kendisinden önceki komutun sonucu ***küçük eşit*** ise gider. CMP AL, BL JLE atla ; AL<=BL ise ***atla*** etiket ismine gider.
JG	Kendisinden önceki komutun sonucu ***büyük*** ise gider. CMP AL, BL JG atla ; AL>BL ise ***atla*** etiket ismine gider.
JGE	Kendisinden önceki komutun sonucu ***büyük eşit*** ise gider. CMP AL, BL JGE atla ; AL>=BL ise ***atla*** etiket ismine gider.
DÖNGÜ KOMUTLARI	
LOOP	CX' teki sayı kadar döngü işlemi yapar. Her döngüde CX bir azalır. CX=0 olunca döngüden çıkar. MOV CX, 05 yap: } 5 kez AL'ye 1 eklenir. Her ADD AL, 1 } CX 1 azalır. CX=0 olunca döngüden çıkılır LOOP yap } LOOP komutundan sonraki satıra geçilir.
ALT PROGRAM KOMUTLARI	
CALL	Alt program çağırma komutu. CALL alt_program_1 **;**Bu komut ile *alt_program_1* isimli alt programa gidilir. Alt programdan dönülünce program CALL komutunun altındaki satırdan devam eder.
RET	Program sonlandırma veya alt programdan ana programa dönmek için kullanılır.
INT İŞLEM KOMUTLARI	
MOV AH, 2 INT 21h	DL' deki ASCII kodun karşılığı karakteri ekrana yazar. Bu komuttan önce DL'ye ascii kodu onulmalıdır.
MOV AH, 9 INT 21h	DS:DX' in gösterdiği adresteki bilgiyi, $ işaretini görene kadar ekrana yazar. Mesaj yazdırmak için kullanılır. Mesajların sonlarına $ işareti muhakkak konulmalıdır.

TANIMLAMALAR	
DB	Değişken tanımlamak için kullanılır. Son karakter **$** olmalı Mesaj1 DB '1. Sayıyı giriniz : $' Mesaj2 DB 10, 13, '2. Sayıyı giriniz : $' 10 ascii kodu bir satır alta geçer, 13 ascii kodu satır başı yapar. İkisi peş peşe kullanılırsa kursör bir satır aşağıya satır başına geçer. 10 ve 13 ascii kodunu mesajın başında veya sonunda kullanabiliriz. Mesaj1 DB '1. Sayıyı giriniz : ' Ah, Dh, '$'