

**Kastamonu Üniversitesi**

**Bilgisayar Mühendisliği Bölümü**

**Mikroişlemciler Laboratuvarı**

**Deney Föyleri**

**LAB 1**

8086 İşlemcilere Giriş ve Bazı Assembly Programlama Komutları

**Ön Çalışma**

8086 Mikroişlemci Yazmaçları

* CPU da bilgiler geçici olarak yazmaçlarda saklanır. Bilgiler 1 ve 2 baytlık veri veya veri adresi olabilir.
* Farklı yazmaçlar farklı fonksiyonlar için kulanılır.
* Yazmaç’ın ilk harfi hangi amaçla kullanıldığını gösterir.

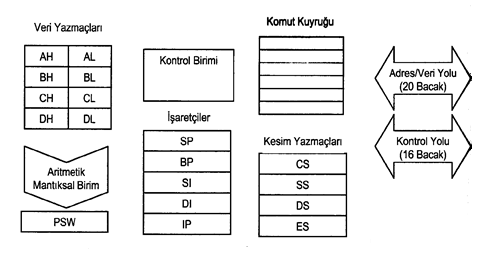
• AX (accumulator)

• BX (base addressing register)

• CX (counter loop operations)

• DX (data in I/O operations)

* İşlemler 3 temel grup altında toplanmış yazmaçlar kullanılarak gerçekleştirilir. Bu yazmaçlar veri(data), kesim(segment) ve işaretçiler(pointer) olarak gruplandırılmıştır.
* Şekil 1.1’de 8086 işlemcisinin basitleştirilmiş iç yapısını görebilirsiniz.

Şekil 1.1

* Veri yazmaçları 16 ve 8 bit olarak kullanılabilir, diğerleri ise sadece 16 bit olarak kullanılır.
* AX yazmaç’ı 16 bitlik bir yazmaçtır ama duruma göre AH ve AL olarak 2’ye bölünerek 8 bitlik yazmaçlar halinde kullanılabilir. Aynı durum BX, CX ve DX yazmaçları için de geçerlidir.
* Yazmaçların bitleri azalan sırayla numaralandırılır.

8 bit yazmaçlar:



16 bit yazmaçlar:



Assembly Dili Programlama Komutları

MOV Komutu(Veri aktarım komutu)

MOV hedef , kaynak; Kaynağı hedefe taşır .Kaynak değeri değişmez.



* Değerler CS, DS, SS ve ES yazmaçlarına doğrudan yüklenemezler.
* CS ve IP yazmaçlarına değer atanamaz.
* İki kesim yazmaç’ı arasında değer aktarılamaz.(önce genel amaçlı yazmaç’a aktarılmalıdır)
* Yazmaçlara kapasitesinden yüksek değerde değer atanamaz.

MOV BX,2H ; BX=0002H, BL:02H, BH:00H

MOV AL,123H ; Hatalı. AL yazmacı 1 baylık değer alabilir.

MOV AX,3AFF21H ; Hatalı. AX yazmacı 2 baytlık değer alabilir.

Örnekler

MOV AX, 0B800h ; AX = B800h (VGA bellek).

MOV DS, AX ; AX’in değerini DS’ye kopyalar.

MOV CL, 'A' ; CL = 41h (ASCII kodu).

MOV CH, 01011111b ; CL = color attribute.

MOV BX, 15Eh ; BX = position on screen.

MOV [BX], CX ; w.[0B800h:015Eh] = CX.

Aritmetik Komutlar

ADD Komutu

ADD hedef , kaynak; Kaynağı hedefe ekler.



Örnekler

MOV AL, 5 ; AL = 5

ADD AL, -3 ; AL = 2

MOV AL,24H ;24H değerini AL’ye taşı

MOV DL,11H ; 11H değerini AL’ye taşı

ADD AL,DL ;AL=AL+DL (AL=35H) (DL =11H)

MOV CH,24H

ADD CH,11H ; CH=35H

SUB Komutu

SUB hedef , kaynak; Kaynağı hedeften çıkarır.



Örnekler

MOV AL, 5 ; AL = 5

SUB AL, 2 ; AL = 3

MOV AL,24H ;

MOV DL,11H ;

SUB AL,DL ;AL=AL-DL

MOV CH,24H ;

SUB CH,11H ; CH=13H

MUL Komutu

MUL operand ; AX = AL \* operand.(operand 1 baytlık olduğu zaman)

;(DX AX) = AX \* operand.(operand 2 baytlık olduğu zaman)

Örnekler

MOV AL, 200 ; AL = 0C8h

MOV BL, 4 MUL BL ; AX = 0320h (800)

DIV Komutu

DIV operand ; AL = AX / operand (operand 1 baytlık olduğu zaman)

AH= Kalan

; AX=(DXAX)/operand (operand 2 baytlık olduğu zaman)

DX=Kalan

Örnekler

MOV AX, 203 ; AX = 00CBh

MOV BL, 4

DIV BL ; AL = 50 (32h), AH = 3

Soru: 0100:1000h ve 0100:2000h bellek adreslerine 34h değerini yazan Intel x86 programını oluşturunuz.

Cevap: org 100h

mov ax,100h

mov ds,ax

mov [1000h],34h

mov [2000h],34h

ret

Soru: İşaretsiz FEh sayısı ile 10h sayısını çarpan ve sonucu 0100:0400h bellek adresine yazan Intel x86 programını oluşturunuz.

Cevap: org 100h

mov ax,100h

mov ds,ax

mov al,0feh

mov bl,10h

mul bl

mov [400h],ax

ret

Soru: CL ve DL yazmaçlarına rasgele bir değer verin. Daha sonra CL yazmacındaki değeri DL yazmacına, DL yazmacındaki değeri ise CL yazmacına yazan Intel x86 programını oluşturunuz.

Soru: 0100:0500h bellek adresindeki 98h değeri ile 0100:0501h adresindeki 52h değerini toplayan ve sonucu 0100:0502h bellek adresine yazan Intel x86 programını oluşturunuz.

Soru: İşaretsiz ABh(171d) sayısını 0Ah(10d) sayısına bölen Intel x86 programını oluşturunuz. Daha sonra bölüm’ü 0100:0500h ve kalanı ise 0100:0502h adresine yazınız.

**LAB 2**

8086 Assembly Programlama Komutları

**Ön Çalışma**

Assembler işletilen komutla birlikte adres uzaklıklarını iki operandla destekler. Bunlar short(kısa), near(yakın) ve far (uzak) operandlarıdır. Short adresi, -128 byte gerideki ve +127 bayt ilerideki bir adresi gösterebilir. Bu uzaklıklardan daha ileride olan bir adres ya near ya da far operandı ile ifade edilir. Near operandı, aynı segment içerisinde -32768 ile +32767 arasındaki byte uzaklıklarında kullanılırken, far operandı diğer bir segmennteki bir prosedürün çağrılmasında ya da 32 KB ve üzeri uzaklıklardaki prosedürlerin çağrılmasında kullanılır.

Döngü Komutları

LOOP Komutu

Bu komut dallanma işlemlerinde sayaç olarak kullanılan CX yazmacının üzerinde bulunan değerin bitip bitmediğini kontrol eder. Program içerisinde bir işlem birden fazla yaptırılacaksa, CX yazmacı o sayı kadar yüklenir. Yani bir döngünün son komutunu oluşturur. Her döngüde sayaç otomatik olarak bir azaltılır ve sorgulanır. Eğer sayaç sıfıra erişmediyse döngüye giren komutlar tekrar işletilir ve sayaç sıfır olana dek işlem tekrarlanır. Eğer sayaç değeri sıfıra eriştiyse döngü bitirilir ve sıradaki program komutlarının işlenmesine devam edilir. LOOP komutunun dönüş adresi +127 ile -128’i geçmemelidir.

Aritmetik Komutlar

DEC Komutu

Tek işlenenli bir komuttur ve işlenenin değerini 1 azaltır.

INC Komutu

Tek işlenenli bir komuttur ve işlenenin değerini 1 arttırır.

Soru: 1’den 5’e kadar olan sayıların toplamını yapan Intel x86 programını oluşturunuz.

Cevap: org 100h

mov AX, 0

mov CX, 5

etiket1:

add AX, CX ; 1’den 5’ekadar olan sayıların toplamını yapar.

loop etiket1

ret

Soru: 1’den 100’e kadar olan sayıların toplamını yapan ve sonucu 0100:1000h bellek adresine yazan Intel x86 programını oluşturunuz.

Soru: 0’dan 5’ekadar olan sayıların faktöriyellerin toplamının 10’a bölümünden kalan sayıyı bulan bir program yazınız.

**LAB 3**

Yığın Kesimi, Bayraklar, Şartlı ve Şartsız Dalma Komutları ve Diğer Komutlar

**Ön Çalışma**

Kesim Yazmaçları

Yığın(Stack) Kesimi

* Program tarafından kullanılan yığın(stack) alanı yığın kesimini oluşturur.
* Yığın kesiminin başlangıç adresi SS yazmacı ile belirlenmektedir.
* Bu kesim, işletim sistemine dönebilmek için gerekli dönüş parametrelerinin saklanması ve programın çalışması sırasında kullanılan PUSH komutlarının kullanımı sonucu oluşan geçici veri saklama ihtiyacını karşılamak üzere ayrılmıştır.
* Yığına son giren ilk çıkar(LIFO-Last In First Out) kuralına göre çalışır.
* Bu kesim içine konulacak verinin nereye yerleştirileceğini veya nereden alınacağını belirleyen işaretçiye SP(Stack Pointer) adı verilmektedir.
* Yığın büyük adresten küçük adrese doğru ilerlediği için PUSH komutu ile yığına bir Word büyüklüğünde veri yerleştirildiğinde SP yazmacının değeri kendiliğinden 2 byte azalacaktır.
* Benzer şekilde yığından POP komutu ile bir word büyüklüğünde veri alındığında ise SP yazmacının değeri kendiliğinden 2 byte artacaktır.
* SP yazmacı, 8086 işlemcisinde yığın kesiminin en üstündeki dolu yeri gösterir.
* SS(yığın kesimi) ve SP(yığın işaretçisi) yığına erişmek için bellekte yüklü olması gerekir.
* CPU’daki her yazmaç(SS ve SP hariç) yığına saklanabilir veya yığından yüklenebilir.
* Sınırlı sayıda yazmaç olduğu için CPU bu depolama alanlarına ihtiyaç duyar.

Yığına İtme(Pushing)

Yığında CPU yazmaçlarına saklama işlemi PUSH olarak adlandırılır.

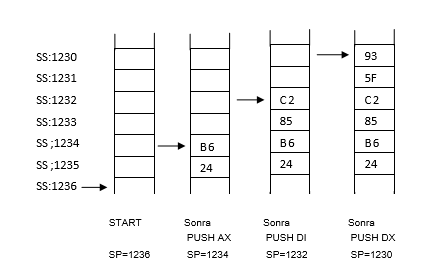
Örnek: SP(STACK POINTER)=1236, AX=24B6, DI=85C2, ve DX=5F93, Aşağıdaki komutlar çalıştığında yığının durumu nasıl olur?

PUSH AX

PUSH DI

PUSH DX

Cevap:



* 8086’da alt byte küçük adreslerde saklanır.

Yığına Çekme(Popping)

Yığında CPU yazmaçlarından yükleme işlemi POP olarak adlandırılır.

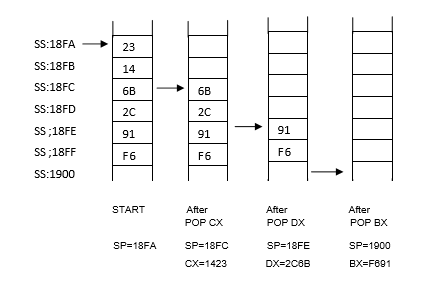
Örnek: SP=18FA, Aşağıdaki komutlar çalıştığında yığının durumu nasıl olur?

POP CX

POP DX

POP BX

Cevap:



BAYRAKLAR(FLAGS)

İşlemci üzerinde veri, işaret, sıra ve kesim grubu dışında bir yazmaç daha mevcuttur. Bu yazmaç, üzerinde bayrak(flag) olarak adlandırılan birer bit’lik özel değerleri tutmaktadır. Bayraklar bazı işlemlerin sonucunda değer değiştirerek sonucu yansıtırlar. 8086 işlemcisinde dokuz adet bayrak vardır. Bayraklar bir bit ile ifade edilmeleri sebebiyle 0 veya 1 değerini alabilirler. Bayrağın değerinin 1 olması durumu Set(birleme), 0 olması durumu ise Clear(silme) veya Reset(ilk durumuna döndürme – sıfırlama) olarak adlandırılır.

* CF: Carry Flag

8’li ve 16’lı işlemlerde oluşan elde/ödünç durumunu kullanıcıya belirtmek üzere CF bayrağının değeri 1 yapılır.

* PF: Parity Flag

İşlem sonucunda çift eşlik(even parity) var ise8yani 1’lerin sayısı çift ise) durumu kullanıcıya belirtmek için PF bayrağının değeri 1 yapılır. Eğer bit dizilişindeki 1’lerin sayısı tek ise(odd parity), PF bayrağının değeri 0 olur.

* AF: Auxilary Carry Flag

8’li işlemlerde düşük anlamlı 4’lüden(nibble) yüksek anlamlı 4’lüye elde/ödünç, 16’lı işlemlerde düşük anlamlı 8’liden yüksek anlamlı 8’liye elde/ödünç aktarılması durumunu kullanıcıya belirtmek üzere AF bayrağının değeri 1 yapılır.

* ZF: Zero Flag

İşlem sonucunun sıfır çıkması(eşitlik olması) durumunu kullanıcıya bildirmek üzere ZF bayrağının değeri 1 yapılır.

* SF: Sign Flag

İşlem sonucunda sayının negatif olduğunu(en anlamlı bit 1 ise) kullanıcıya belirtmek üzere SF bayrağının değeri 1 yapılır.

* TF: Trap Flag

Adım bayrağı olarak adlandırılır. TF bayrağının 1 değeri alması durumunda her komut yürütüldükten sonra bir kesme üretilerek durumun kullanıcı tarafından izlenmesine imkan sağlanır(single-step mode).

* IF: Interrupt Flag

IF, maskelenebilir kesme(maskable interrupt) isteklerinin izinli olup olmadığını belirlemek üzere kullanılmaktadır. IF bayrağının değeri 1 ise kesmelere izin verecektir. IF bayrağının sıfır olması durumunda INT komutu kullanılarak üretilen her türlü maskelenebilir kesme devre dışı olacaktır.

* DF: Direction Flag

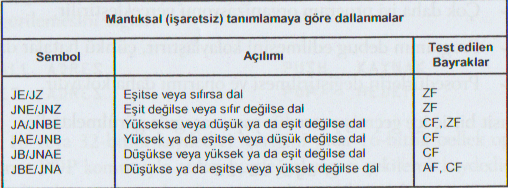
Dizgi işlemlerinde işlemin gözüne yüksek adresli bellek gözünden başlayarak düşük adresli belleğe doğru yürütülebilmesi için DF bayrağının 1 değerini alması gereklidir. Aksi halde işlem düşük adresten başlayarak yüksek adrese doğru gerçekleşecektir.

* OF: Overflow Flag

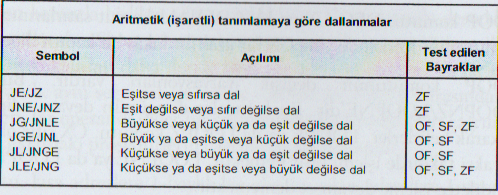
Aritmetik taşma olması halinde durumu kullanıcıya belirtmek üzere OF bayrağının değeri 1 yapılır. Aritmetik taşma işaretli sayılar ile yapılan işlemlerde elde edilen işaretli sonucun tanımlı alana sığmaması(taşması) durumunu belirlemektedir.

Şartlı Dalma Komutları

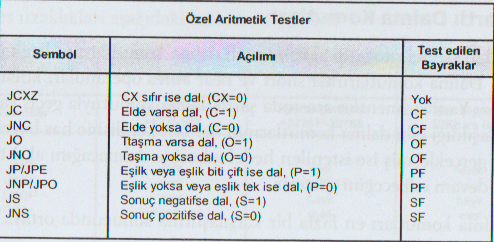
Mikroişlemci program akışını şartlı dalma komutlarının bağlı olduğu bayraklara göre değiştirir. Komutların her biri kendine has bayrakları kullanır. Eğer bu bayraklar gerçekleşmiş ise istenilen hedef adresine dalınacağını aksi taktirde normal program akışına devam edileceğini gösterir. Dalma komutlarında short ve near adres operandları kullanılırken, far operandı kullanılamaz. Yani kesimler arasında şartlı dalma komutuyla geçiş yapılamaz. Şekil 3.1, Şekil 3.2 ve Şekil 3.3’de çeşitli koşullu dalma komutları, açılımı ve test edilen bayraklar verilmiştir.



Şekil 3.1



Şekil 3.2

 Şekil 3.3

Şartsız Dalma Komutları

“JMP label” program akışının koşulsuz olarak hedef etikete geçtiği atlamadır.

Aşağıdaki gibi gösterimleri vardır.

1. KISA ATLAMA, “JMP SHORT label” formatıyla gösterilir. Hedef adres o anki mevcut IP’nin 128 ile +127 aralığında olabilir

2. YAKIN ATLAMA, “JMP label”. Mevcut Kod Kesimi’nin içinde atlar. Aynı kısa atlama gibi ama farkı hedef adres o anki mevcut IP’nin +32767 ile–32768 aralığında olabilir.

3. UZAK ATLAMA, “JMP FAR PTR label” formatıyla gösterilir. Burada hem Kod Kesim değeri hem de IP değeri değişir.

Çağırma Komutları

CALL Komutu

CALL Etiket #Mevcut adresi yığına atar(PUSH) ve Etiket’ e atar

.

.

Etiket:

.

.

Ret #Ret komutu geldiğinde yığına atılan adresi yığından çeker ve o adrese gider.

Not: Etiketten sonra herhangi bir PUSH veya POP işlemi yaparsak yığın değişeceği için ret komutu geldiğinde program yanlış adrese geri dönecektir.

Aritmetik Komutlar

CMP Komutu

CMP operand1 - operand2

İşlemleri karşılaştırmak için kullanılan bir komuttur. Sonuçlar yazılmaz sadece sonuca göre bayrak yazmaçları değişir.(OF, SF, ZF, AF, PF, CF)

Örnek:

MOV AL, 5

MOV BL, 5

CMP AL, BL ; AL-BL=0 olduğu için , ZF = 1 olacak

RET

Soru: 1000 ve 2000 adresli bellek gözlerindeki iki sayıyı toplayan ve sonuç FFH’tan büyükse 3000 adresine 99h, küçükse 22h yazan programı oluşturunuz.

Cevap: Org 100h

mov ax, 0h

mov [1000], 22h

mov [2000], 88h

mov al, [1000]

add al, [2000]

jc buyuk

mov [3000], 22h

jmp stop

buyuk: mov [3000], 99h

stop: ret

Soru: Aynı soruyu verilen adreslerin onaltılık tabana çevrilmiş haliyle tekrar yazınız.

Soru: 1300h ile 130Eh adresleri arasında tüm verileri toplayan ve sonucu 2200h’ye ve oluşan eldeyi de 2201h’ye yazan programı oluşturunuz.

Soru: 100’den küçük 3’e ve 2’ye tam bölünebilen doğal sayıların toplamını bulan ve bu toplamı DX yazmacın yazan programı yazınız.

**LAB 4**

Soru: 1000h – 1001h adresli bellek gözlerindeki iki sayıyı çarpan, çarpım sonucunda elde oluşmuşsa oluşan eldeyi 1002h adresine, işlem sonucunu da 1003h’ye yazıp duran programı yazınız.

Cevap: Org 100h

mov [1001h], 12h

mov [1000h], 21h

cmp [1001h], 0

jne carp; zf=0 ise

jmp son

carp: mov al, [1000h]

mov bl, [1001h]

mul bl

jc elde

jmp yaz

elde: mov[1002h], ah

yaz: mov[1003h],al

son: ret

Soru: 1000h adresli bellek gözündeki sayıyı 1001h adresli bellek gözündeki sayıya bölen, bölme sonucunu 1002h adresli bellek gözüne, bölme işlemi sonucunda oluşan sayıyı 1003h adresli bellek gözüne, bölen sayı sıfırsa bölme işleminde hata oluşacağından bu hata durumunu belirtmek için 1004h adresli bellek gözüne 1 sayısını yazan programı yazınız.

Soru: Yığın yapısını kullanarak, bellekteki bir konumdan art arda 10 sayı okuyan ve bu sayıların her adımdaki ortalamasını(toplamda 10 ortalama olmak üzere) yığına koyan bir program yazınız.

**LAB 5**

Adresleme ve Assembly Programlama Komutları

**ÖN ÇALIŞMA**

Program Segmentlerine(Kesim) Giriş

Kesim: 64K byte’a kadar çıkabilen hafıza alanıdır ve 16 ile kalansız bölünebilen bir adres ile başlar

Assembly programlama dili üç kesimden oluşur.

* Kod Kesimi: Program kodunu içerir(talimatlar, komutlar)
* Veri Kesimi: Program tarafından işlenecek olan veriinin(bilgi) saklanmasında kullanılır.
* Yığın Kesimi: Bilginin geçici olarak saklanmasında kullanılır.

Mantıksal ve Fiziksel Adres

* Fiziksel Adres gerçekten adres yoluna koyulan 20 bit adrestir. 00000h-FFFFFh aralığına sahiptir.
* Offset(Göreli Konum) Adres 64K byte kesim aralığında bir konumdur.
* Mantıksal Adres kesim adresi ve göreli konum adresinden oluşur.

Veri Kesiminde Adresleme

* 8086, programı çalıştırmak için komutları veri kesiminden çeker.
* Bir komutun mantıksal adresi CS(kod kesimi) ve IP(komut işaretçisi) den oluşur.





Mantıksal Adres: CS:IP

↘göreli konum adresi

Fiziksel Adres, CS nin bir hex basamak sola kaydırılması ve IP’nin eklenmesi ile üretilir.

Örnek: CS:IP => 2500:95F3H

1. CS ile başla 2500
2. CS’yi sola kaydır 25000
3. IP’yi ekle 2E5F3 (25000+9SF3)

Mikroişlemci 2E5F3 adresinden başlayarak hafıza alanlarından dönüşümlü olarak komutları alacaktır.

Örnek: CS=24F6H ve IP=634AH ise, aşağıdakileri belirleyiniz:

a) Mantıksal adres

b) Göreli konum adres

c) Fiziksel adres

d) Kod kesiminin alt aralığı

e) Kod kesiminin üst aralığı

Cevap:

a) Mantıksal adres; 24F6:634A

b) Göreli konum adres; 634A

c) Fiziksel adres; 24F60+634A= 2B2AA

d) Kod kesiminin alt aralığı: 24F6:0000 => 24F60+0000 =24F60

e) Kod kesiminin üst aralığı: 24F6:FFFF => 24F60+FFFF=34F5F

Veri Kesimi

* Hafızanın kesin bir şekilde veri için ayrılan kısmına veri kesimi (DS) denir.
* Kod kesimi nasıl CS ve IP ile ilişkiliyse, veri kesimi de DS ve bir göreli konum değeri kullanır. 8086 da, BX, SI ve DI göreli konum adreslerini tutmak için kullanılır.

Örnek: DS=7FA2H ve ofset 438EH ise, aşağıdakileri belirleyiniz:

a) Fiziksel adres

b) Veri kesiminin alt aralığı

c) Veri kesiminin üst aralığı

d) Mantıksal adresi gösteriniz.

Cevap:

a) Fiziksel adres; 7FA20+438E= 83DAE

b) Alt aralık: 7FA20(7FA20+0000)

c) Üst aralık: 8FA1F(7FA20+FFFF)

d) Mantıksal adres; 7FA2:438E

Veri Kesiminin Kullanımı

5 bayt veri eklemek için bir programın gerekli olduğunu varsayalım (25H, 12H, 15H,1FH ve 2BH)

Birinci yol: MOV AL,00H ; AL sıfırlanır

ADD AL,25H

ADD AL,12H

ADD AL,15H

ADD AL,1FH

ADD AL,2BH ; AL=25+12+15+1F+2B

Diğer yol: Veri kesimi için göreli konumun 200H den başladığını varsayalım.

DS:0200 = 25

DS:0201 = 12

DS:0202 = 15

DS:0203 = 1F

DS:0204 = 2B

MOV AL,0 ;AL temizlenir

ADD AL,[0200] ;DS:200 in içeriği AL’ye eklenir

ADD AL,[0201] ;DS:201 in içeriği AL’ye eklenir

ADD AL,[0202] ; DS:202 in içeriği AL’ye eklenir

ADD AL,[0203] ; DS:203 in içeriği AL’ye eklenir

ADD AL,[0204] ; DS:204 in içeriği AL’ye eklenir

Küçük sonlu kuralı (Little endian convention)

8-bit (1-bayt) veri göz önüne alındığında, baytlar hafızada birbiri ardına saklanır. Fakat 16-bit (2-bayt) veri göz önüne alındığında veri nasıl saklanır?

Örnek: MOV AX,35F3H : AX’e 35F3H yükle

MOV [1500],AX : AX’in içeriği 1500H ofsetine kopyala

Böyle bir durumda düşük bayt düşük hafıza alanına ve yüksek bayt yüksek hafıza alanına gider.

DS:1500 = F3 DS:1501 = 35

Bu kural küçük sonlu kuralı (little endian convention) olarak adlandırılır: bu kural Intel tarafından kullanılmaktadır. Büyük sonlu kuralı (big endian convention) bunun tam tersidir, yüksek bayt düşük adrese ve düşük bayt yüksek adrese gider. Motorolla mikroişlemsi bu kuralı kullanır.

Yığının mantıksal ve fiziksel adresi

Yığın için fiziksel adresin hesaplanmasında, kod ve veri kesimleri için kullanıldığı gibi, aynı prensip uygulanır. Fiziksel adres yığın kesimi(SS) yazmacı ve yığın işaretçisinin (SP) değerine bağlıdır.

Örnek: SS=3500H ve SP:FFFEH ise;

a) Fiziksel adresin hesaplanması: 35000+FFFE = 44FFE

b) Yığının kesiminin alt aralığının hesaplanması: 35000+0000 = 35000

c) Yığın kesiminin üst aralığının hesaplanması: 35000+FFFF = 44FFF

d) Yığının mantıksal adresi: 3500:FFFE

lAssembler Veri Talimatları

* ORG (Origin) : Offset adresinin başlangıcını göstermek için kullanılır. ORG’dan sonra gelen sayı hex veya decimal olabilir. Çoğunlukla Veri kesiminde kullanılır.
* DB (define byte) : Decimal, binary, hex ve ASCII byte’lar için bellek tahsisine olanak sağlar.
* DUP (duplicate) : Karakterlerin belirli bir sayıda çoğaltılması için kullanılır.
* DW(define word) : Aynı anda bellekte 2 bayt (bir kelime) yer tahsis etmek için kullanılır. DUP karakterlerin belirli bir sayıda çoğaltılması için kullanılır.
* EQU(equate) : Hafızada yer işgal etmeden bir sabit tanımlamak için kullanılır.

Veri Aktarım Komutları

LEA(Load Effective Adress Komutu)

LEA komutu, bellek operandının göreli konum adresini genel amaçlı, işaretçi ve indis yazmaçlarına aktarır.

Örnek: MOV BX, 35h

MOV DI, 12h

LEA SI, [BX+DI] ; SI = 35h + 12h = 47h

XCHG Komutu

Değiş-tokuş komutu kaynak operanla hedef operandın yerini değiştirir.

Örnek: MOV AL, 5

MOV AH, 2

XCHG AL, AH ; AL = 2, AH = 5

XCHG AL, AH ; AL = 5, AH = 2

RET

NOT: XCHG AX,[1234H] →Bellekten yazmaca veya bellekten belleğe olmaz.

ÖTELEME VE DÖNDÜRME KOMUTLARI

SHL KOMUTU

Birinci operandı sola kaydırır. Kaydırma sayısı ikinci operandla belirtilir.

Örnek: Mov al, 11100000b

Shl al,2; al→10000000b cf→1

SHR KOMUTU

Birinci operandı sağa kaydırır. Kaydırma sayısı ikinci operandla belirtilir.

Örnek: Mov al, 11100000b

Shl al,2; al→00111000b cf→0

Yığın Komutları

POPF KOMUTU

Yığından 2 bayt alır bayrak yazmacına atar.

flags = SS:[SP] (top of the stack) SP = SP + 2

PUSHF KOMUTU

Bayrak yazmacını alır 2 bayt olarak yığıta kaydeder.

SP = SP - 2 SS:[SP] (top of the stack) = flags

Soru: Belleğe sayılar etiketi ile tanımlı 10 tane 1 baytlık sayıdan en küçüğünü bulan ve bunu al yazmacına atan programı yazınız.

Cevap:

org 100h

lea si, sayilar ; sayilar etiketi ile tanimli ilk sayinin ofset adresini si yazmacına attim

mov cx, 9 ; 9 kez loop’a sokacağım

mov al, [si] ; si adresine denk gelen sayiyi al yazmacina atiyorum(al min deger tutar)

dongu:

inc si ;bir sonraki adrese gidiyorum

mov bl, [si] ; o adresteki sayiyi bl’ye atiyorum

cmp al, bl ; al ile bl’yi kiyasliyorum

jb devam ; eger al kucukse al degerini degistirmez

mov al, bl ; eger al buyukse kucuk değeri al’ye atar

devam:

loop dongu

hlt

sayilar db 9, 6, 3, 5, 4, 7, 12, 1, 6, 35

**LAB 6**

Mantıksal, Döndürme ve Dizgi Komutları

**ÖN ÇALIŞMA**

Mantıksal Komutlar

NOT

NOT AX

Ör: MOV al,34h

NOT al --> AL =CBH

AND

AND işlenen1 ,işlenen2

Ör: MOV al,10011001b

MOV ah,01100110b

AND al,ah--> AL=00000000b

OR

OR işlenen1 ,işlenen2

Ör: MOV al,10011001b

MOV ah,01100110b

OR al,ah--> AL=11111111b

XOR

XOR hedef, kaynak

Ör: MOV DH,54H

XOR DH,78H

Çözüm: 54H 01010100

78H 01111000

─── ────────

2C 00101100

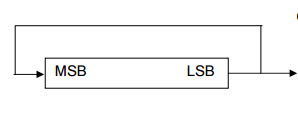
SF=0, ZF=0, PF=0, CF=OF=0

XOR talimatı, bir yazmacın kendisi ile XOR işlemi yapılarak içeriğinin temizlenmesi için kullanılabilir.

Döndürme Komutları

ROR Komutu

İşlem sonucunda işlenen sağa doğru ilerlemektedir. CF bayrağı işlemi doğrudan etkilemez ancak işlem sonunda en az anlamlı bit CF bayrağında olacaktır.



Ör : MOV BX,C7E5H ;BX=1100 0111 1110 0101

MOV CL,6 ;CL=6 döndürme sayısı

ROR BX,CL ;BX=1001 0111 0001 1111 CF=1

Ör: MOV AL,36H ;AL=0011 0110

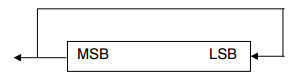
ROR AL,1 ;AL=0001 1011 CF=0

ROR AL,1 ;AL=1000 1101 CF=1

ROR AL,1 ;AL=1100 0110 CF=1

ROL Komutu

Sola doğru yapılan bir döndürme işlemi olmasına rağmen CF bayrağı doğrudan işlemin sonucunu etkilememektedir. Ancak döndürme yapıldıktan sonra işlenenin en anlamlı bit’i CF bayrağına gelmektedir.



Ör:

MOV AL,47H ;AL=0100 0111

ROL AL,1 ;AL=1000 1110 CF=0

ROL AL,1 ;AL=0001 1101 CF=1

ROL AL,1 ;AL=0011 1010 CF=0

DİZGİ KOMUTLARI

SCASB

AL yazmacı ile ES:[DI] belleğindeki byte’ı kıyaslar.

NOT: DF, the Direction Flag: DI, SI registerlarının artma veya azalma modunda olmasını denetler.

AL - ES:[DI]

DF yazmacına göre DI değerini değiştirir. (DF=1 için registerlar azalır, 0 için artar)

SCASW

AX yazmacı ile ES:[DI] belleğindeki word’ ü kıyaslar.

LODSB

DS:[SI] deki değeri AL’ye at, SI’yı güncelle.

DF yazmacına göre SI değerini değiştirir. (DF=1 için registerlar azalır, 0 için artar)

LODSW

DS:[SI] deki değeri AX’e at, SI’yı güncelle.

STOSB

AL değerini ES:[DI] bellek adresine yükler. DI yı günceller.

DF yazmacına göre DI değerini değiştirir. (DF=1 için registerlar azalır, 0 için artar)

STOSW

AX değerini ES:[DI] bellek adresine yükler. DI yı günceller.

MOVSB

DS:[SI] den ES:[DI] ya kopyalar. SI ve DI günceller.

DF yazmacına göre DI değerini değiştirir. (DF=1 için registerlar azalır, 0 için artar)

MOVSW

MOVSB komutunun yaptığı işi word sınırında yapan komuttur.

REP

MOVSB, MOVSW, LODSB, LODSW, STOSB, STOSW komutlarını CX kere tekrarlatır.

REPE

CMPSB, CMPSW, SCASB, SCASW komutlarını en fazla CX kere tekrar ederek ZF=1 iken (sonuç eşitse) programı bitirir.

REPNE

CMPSB, CMPSW, SCASB, SCASW komutlarını en fazla CX kere tekrar ederek ZF=0 iken (sonuç eşit değilse) programı bitirir.

REPNZ

CMPSB, CMPSW, SCASB, SCASW komutlarını en fazla CX kere tekrar ederek ZF=0 iken (sonuç 0 değilse) programı bitirir.

REPZ

CMPSB, CMPSW, SCASB, SCASW komutlarını en fazla CX kere tekrar ederek ZF=1 iken (sonuç =0 ise) programı bitirir.

Örnek: Bir bayttaki 1’lerin sayısını bulan programı yazınız.

Cevap:

org 100h

sub bl,dl ; 1’lerin sayısını tutmak için BL’yi temizle

mov dl,8 ; tamamini 8 kez dondur.

mov al,97h

again: rol al,1 ; bir kez dondur

jnc next ; 1 icin kontrol et

inc bl ; CF=1 ise sayaç arttir

next:

dec dl ; 8 kez uzerinden git

jnz again; bitmediyse geri don

hlt

Soru: BCDEh değerini 0100:0300h ve 0100:03FFh bellek adresleri arasına yükleyen programı yazınız.

Cevap:

org 100h

mov ax, 0100h

mov es, ax

mov di, 0300h

mov ax, 0BCDEh

mov cx, 128

rep stosw

ret

Soru: Aşağıdaki programı yorumlayınız.

org 100h

mov ax, cs

mov ds, ax

mov es, ax

cld

mov cx, 14

lea si, YAZI

mov di, 200h

DEVAM:

LODSB

STOSB

LOOP DEVAM

HLT

ret

yazi db ‘’Merhaba Dunya!’’

**LAB 7**

Ekran İşlemleri

**Önçalışma**

INT 10h / AH = 0 – video türünü ayarla.

INT 10h / AH = 01h – imleç şeklini ayarla

CH = başlama satırı

CL = bitiş satırı

INT 10h / AH = 2 – imleç konumunu ayarla

girdi:

DH = y kordinatı

DL = x kordinatı

BH = sayfa no (0..7).

Örnek:

mov dh, 10

mov dl, 20

mov bh, 0

mov ah, 2

int 10h

INT 10h / AH = 03h – imlecin konumunu ve şeklini al.

girdi: BH = sayfa no.

çıktı:

DH = y kordinatı.

DL = x kordinatı.

CH = imlecin başladığı satır.

CL = imlecin biteceği satır.

INT 10h / AH = 05h – görüntü sayfasını ayarla

girdi:

AL = görüntü sayfası (0..7).

INT 10h / AH = 06h – temizle ve ekranı yukarı kaydır.

INT 10h / AH = 07h – temizle ve ekranı aşağı kaydır.

girdi:

AL = kaç satır yukarı ya da aşağı (00h = ekranı temizle).

BH = görüntü özelliği

CH, CL = sol üst köşe x ve y kordinatları.

DH, DL= sağ alt köşe x ve y kordinatları.

INT 10h / AH = 08h – karakter okuma ve imleç konumu

girdi:

BH = sayfa no.

çıktı: AH = karakter özelliği.

AL = karakter.

INT 10h / AH = 09h – karakter yaz ve imleç konumu

girdi:

AL = yazılacak karakter.

BH = sayfa no.

BL = karakter özelliği.

CX = tekrar sayısı.

INT 10h / AH = 0Ah – imleç konumundaki karakteri yaz.

girdi:

AL = yazılacak karakter.

BH = sayfa no.

CX = tekrar sayısı.

INT 10h / AH = 0Bh – kenarlık değerlerini ayarla.

INT 10h / AH = 0Ch – pikselin rengini değiştir.

girdi: AL = piksel rengi

CX = x

DX = y.

örnek:

mov al, 13h

mov ah, 0

int 10h ; video türünü ayarla

mov al, 1100b

mov cx, 10

mov dx, 20

mov ah, 0ch

int 10h ; pikselin rengini değiştir.

INT 10h / AH = 0Dh – pikselin rengini atar.

girdi:

CX = x.

DX = y.

çıktı: AL = piksel rengi

INT 10h / AH = 0Eh – karakter yazdırır.

girdi:

AL = yazılacak karakter

örnek: mov al, 'a'

mov ah, 0eh

int 10h

INT 10h / AH = 13h – dizgi yazdırır.

girdi:

AL = yazı tipi.

BH = sayfa no.

BL = özellik

CX = dizgi uzunluğu

DL = başlama sütunu

DH = başlama satırı

ES:BP yazılacak dizgi

örnek:

mov al, 5

mov bh, 0

mov bl, 0011\_1011b; sol 4 bit arka plan, sağ dört bit yazı rengi

mov cx, msg1end - offset msg1 ; mesaj boyutunu hesapla

mov dl, 10

mov dh, 7

push cs

pop es

mov bp, offset msg1

mov ah, 13h

int 10h

jmp msg1end

msg1 db " hello, world! "

msg1end:

Renk tablosu:

HEX BIN COLOR

0 0000 black

1 0001 blue

2 0010 green

3 0011 cyan

4 0100 red

5 0101 magenta

6 0110 brown

7 0111 light gray

8 1000 dark gray

9 1001 light blue

A 1010 light green

B 1011 light cyan

C 1100 light red

D 1101 light magenta

E 1110 yellow

F 1111 white

SORU: Bellekte dikdörtgen db 4,5 şeklinde 2 baytlık sayı bulnmaktadır. İlk sayı yatay, ikinci sayı dikey uzunluğu belirtecek şekilde ‘\*’ karakterlerinden oluşan bir dikdörtgen çizin.

Cevap:

org 100h

lea si, dikdörtgen

mov cx, 0

mov cl, [si]

mov dl, [si+1]

devam:

mov al,’\*’

mov ah, 0Ah

mov bx, 0

int 10h

mov ah, 0eh

mov al, 0ah; satir atlar

int 10h

dec dl

cmp dl,0

jne devam

ret

dikdörtgen db 4,5