Deney No : 3

**Deney Tarihi :** 05.11.2015

**Deneyin Amacı :** Yığın(Stack) Kesimi, Yığına İtme(Pushing), Yığına Çekme(Popping), Bayraklar(Flags), Şartlı ve Şartsız Dalma Komutları, Çağırma Komutları, Aritmetik Komutları Kullanarak Örnekleri Çözmek

**SORU VE CEVAPLAR**

Soru-1: Aşağıdaki soruyu verilen adreslerin onaltılık tabana çevrilmiş haliyle tekrar yazınız.

1000 ve 2000 adresli bellek gözlerindeki iki sayıyı toplayan ve sonuç FFH’tan büyükse 2500 adresine 99h, küçükse 22h yazan programı oluşturunuz.

Cevap-1:  
  
**Org 100h**

**mov ax,0h :** ax yazmacının değerini sıfırlıyoruz.

**mov [3e8h],22h :** 3e8h bellek adresine 22h değerini atıyoruz.

**mov [7d0h],88h :** 7d0h bellek adresine 88h değerini atıyoruz.

**mov al, [3e8h] :** al yazmacına 3e8h adresindeki değeri atıyoruz.

**add al, [7d0h] :** al yazmacına 7d0h adresindeki değer ile ekleme yaparak al değerine yazıyor.

**jc buyuk :** CF bayrağında elde varsa buyuk etiketine gönderiyoruz.

**mov [9c4h], 22h :** 9c4h bellek adresine 22h değerini atıyoruz.(bir üst durumda eğer elde yok ise bu adımı yapıyor)

**jmp stop :** programı sonlandırmak için stop adımına gönderiyoruz.

**buyuk: mov [9c4h], 99h :** 9c4h bellek adresine 99h değerini atıyoruz.(eğer elde var ise bu adıma gelerek bu işlemi yapıyor)

**stop: ret :** programı sonlandırıyor

Soru-2: 1300h ile 130Eh adresleri arasında tüm verileri toplayan ve sonucu 2200h’ye ve oluşan eldeyi de 2201h’ye yazan programı oluşturunuz.

Cevap-2:

**org 100h**

**mov ax,0h** : ax yazmacını sıfırlıyoruz.

**mov sp,130eh** : sp yazmacına 130eh değerini atıyoruz.

**mov cx,7** : cx yazmacına 7 değerini atıyoruz. (Döngünün sayısı)

**etiketyaz:** : etiketyaz yazma döngümüz. Döngü sayısı kadar işlem yapacak.

**push cx** : 130eh adresinden itibaren ekleme işlemi yapılıyor.

**loop etiketyaz** : Döngüdeki cx yazmaç değeri birer azaltılarak etiketyaz’a gidiyoruz. cx değeri 0 olana kadar işlem yapılıyor.

**mov cx,7** : cx yazmacına tekrar 7 değerini atıyoruz.(Döngünün sayısı)

**etiketoku:** : etiketoku okuma döngümüz. Döngü sayısı kadar işlem yapacak.

**pop bx** : 1300h adresinden itibaren değerler okunarak bx yazmacına aktarılıyor.

**add ax,bx** : okunan değerler ax yazmacına eklenerek sonuç bulunuyor.

**loop etiketoku** : cx yazmacındaki değer 0 olana kadar etiketoku’ya giderek işlemleri yapıyor.

**mov [2200h],ax** : topladığımız değerlerin sonucunu yani ax yazmacındaki değeri 2200h adresine atıyoruz.

**ret** :

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

Cevap-3:

**org 100h**

**mov cx,100** : cx yazmacına 100 değerini atıyoruz.

**dongu:** : Bölümler için döngü oluşturuyoruz.

**call ikibol** : İkiye bölme yapıyoruz.

**atla:** : Kalan sıfırsa atlamak için başka bir döngü oluşturuyoruz.

**loop dongu** : cx yazmacındaki değer 0 olana kadar dongu etiketine gidiyor.

**hlt** : Programı sonlandırıyoruz.

**ikibol:** : cx yazmacındaki değerin 2’ye bölünüp bölünemediğini kontrol ediyoruz.

**mov ax,cx** : cx yazmacındaki o anki sayıyı ax yazmacına atıyoruz.

**mov bl,2** : bl yazmacına 2 değerini atıyoruz.

**div bl** : ax yazmacındaki değeri 2’ye bölüyoruz.

**cmp ah,0** : kalan 0 mı diye kontrol ettiriyoruz.

**jne atla** : eğer 0’a eşitse alttaki komutları çalıştır.

**ucbol:** : cx yazmacındaki değeri 3’e bölünüp bölünmediğini kontrol ediyoruz.

**mov ax,cx** : cx yazmacındaki değeri tekrar ax yazmacına atıyoruz.

**mov bl,3** : bl yazmacına 3 değerini atıyoruz.

**div bl** : ax yazmacındaki değeri bl değeri yani 3 bölüyoruz.

**cmp ah,0** : kalan 0 mı diye kontrol ediyoruz.

**jne atla** : kalan 0 değilse loop’a dönmesini sağlıyoruz.

**add dx,cx** : sonuç 0’sa cx’deki değeri dx’e atıyoruz.

**ret** : programı sonlandırıyoruz