Her programlama dilinde oldugu gibi assembly programlama dilinde de kod akisi yukaridan asagiya dogrudur.

Ucuncu satirda Arduino UNO uzerinde B kismi (8-13 port arasi) sadece on ucuncu port uzerinden anot ozelligini bu portlara vermek istedik ve bu olayi OUT komutu sagliyor. Bu arada on ucuncu port , Arduino UNO uzerinde dahili LED’in portudur. Ikinci satirda ise B kisminda olan butun portlarin varsa anot ozelliklerini sifirlamasini istedik.

Diger programlama dillerinden bagimsiz olarak assembly programlama dilinde loop , delay , L1 gibi komutlar fonksiyon gibi davransa da cagirilmasa dahi kod akisina gore icerisine girip komutlari calistirir. Takma ad almalarinin sebebi yine fonksiyonlarin davranisi gibi tekrar cagirilmak istendiklerinde takma adlarina gore cagirilabilirler. Bu yuzden bunlarin ismine label denir.

Loop labeli icerisine girdik. Dikkatimizi ceken olay CALL komutu ile delay labelini cagiriyor olmamizdir. Cunku LED’in yanip sonmesini istiyoruz ve bunu gozlemleyebilmek icin bir gecikme suresi koymak gerekir. Bu delay yirmi ucuncu , dorduncu ve besinci yazmaclardaki degerlerle orantilidir. Su anki gecikme suresi uc saniyedir ve her uc saniyede bir LED’in durumu degisecektir. Delay labeli icerisinde yazmaclara degerler atandiktan sonra L1 labelindan devam edip bu label icerisinde degerler azaltilip eger degerlerden bazilari sifira esit degilse tekrardan bu labela geri donuyor ve bu olay BRNE komutu ile yapilir. L1 icerisinde islem tamamlandigi zaman RET komutu ile geri donuluyor ve butun surecin tekrar etmesi icin RJMP loop yazilarak donguye aliniyor.