Ilk olarak tekrar tekrar yazmamamiz icin on altinci ve on yedinci yazmaclara def komutuyla sabit degerler vererek kaydettik. Ustten alta dogru kod siralamamizda bizi start label i karsiliyor. Start label inda baslangic olarak on altinci yazmactaki degerimizin bir olmasini ve asagida ise PORTB dedigimiz icin sekizinci porttan anot olarak davranis gostermesini istiyoruz. Sonrasinda on yedinci yazmaca baslangic degeri olarak yuz degerini veriyoruz. CALL komutu ile EEPROM yazdirma ve okuma islemlerini yazdigim label icerisine cagirip tekrardan cagirilan kisma gitmesini istedigim icin bu label larin sonuna ret (return) yani geri donmesini saglayacak komutu yazdik.

Start label i bittikten sonra loop label ina gecis yapiyoruz. Baslangicta PIND diye bir ifade bizi karsiliyor. D kismi , 0 – 8 port arasinda olan kisimdir. SBIC komutu , eger atanan port tetiklenmisse kendisinden sonraki satira atla demektir. Ikinci pindeki butona bastigimiz zaman nopressbutton1 label ina gitme komutunu atlayacaktir. Butonlar uzerinde basildigi zaman bir geri tepme olayi oldugu icin gecikme koymamiz gerekiyor. Butona basildigi zaman devre uzerinde gelen sinyal direk sifirdan bire direkt degismez , bu sirada bir dalgalanma olusturur. Ikinci pinde olan butonda EEPROM icerisindeki degeri artirirken , ucuncu pindeki buton ise azaltip EEPROM icerisine yazdirmaktadir.

Artirma ve azaltma islemlerini yaptiktan sonra asil gorevimiz olan eger EEPROM icerisindeki sayi cift ise LED in yanmasini tekse yanmamasini istiyorduk. Assembly dilinde oldugumuz icin bu islemi AND operatoruyle sagliyoruz. EEPROM uzerinden gelen sayiyi bir sayisiyla AND operatoruyle islem yaptiriyoruz. Mesela EEPROM’daki deger yuz olsun ve bir sayisiyla isleme soktugumuz zaman sonuc olarak bitlerindeki degerlerin hepsi sifir gelir ve bu da degerimizin cift oldugunun gostergesidir. Bu sefer de yuz bir geldigini dusunelim , sonuc olarak en sagdaki bit haricinde butun bit degerlerinin sifir geldigini fakat en sagdaki bit degerinin bir geldigini gorebilir yani sayimizin tek oldugunu aciklar. BREQ komutu ise eger kendisinden onceki ifade sifira esitse komuttan sonra label a gitmektedir yani sayimiz cift ise LED in yanmasi icin gereken komutlarin oldugu label a gitmektedir. is\_even label inda ise sekizinci pinde anot ozelligi kazandirip LED imizin anot pinini bu pine bagliyoruz.

Diyelim ki ikinci pinde olan butona degil de ucuncu pinde olan butona bastigimizin senaryosunu kuralim. loop label inda sbic PIND , 2 kisminda ikinci pinde olan butonu tetiklemedigimiz kendisinden sonraki satiri atlamayacagi icin nopressbutton1 label ina gidecektir. nopressbutton1 label inda ise loop label inda oldugu gibi ayni komutlar yer almaktadir. Baslangicta ucuncu pinde yer alan butonun tetiklenip tetiklenmedigini kontrol ediyoruz. Eger bu butona da basilmazsa nopressbutton2 label inda yer alan jmp loop yani tekrardan loop label ina geri donmesini sagliyoruz. Eger tetiklenirse kendisinden sonra gelen satiri atlayip gecikme alacaktir. Bu butondaki olay , EEPROM’daki veriyi bir azaltip EEPROM icerisine tekrardan yazdirmaktir. Yazdirdiktan sonra yazilan degerin tek veya cift olmadigi AND operatoruyle kontrol edilir. Eger ciftse is\_even label ina gider ve sekizinci pinde anot ozelligi kazandirir . Eger tekse altindaki satirlara atlayip B kisminda olan portlarin hepsine sifir bitini yollayip LED imizi inaktif yapar.