KOMUT SATIRI	AÇIKLAMA
ORG 00H	bu satırdan sonraki kodlar 00H adresinden başlasın
SJMP MAIN	MAIN etiketine atla
ORG 03H	bu satırdan sonraki kodlar 03H adresinden başlasın bu adres INT0(Harici Kesme 0) kesmesinin çalıştığı adrestir
LJMP START	START etiketine atla burada her bir kesmenin çalışacağı kod bloğu 8 bayt ile sınırlıdır(her kodun bayt olarak kapladığı alan farklıdır yani kod 1 satır 1 bayt yer kaplıyor olarak düşünülmesi yanlıştır) INT0 kesmesi için yazacağım kod 8 bayttan daha uzun olabilme ihtimalini düşünerek çalışmasını istediğim kodları ayrı bir yerde yazıyorum ve bu kodları yazdığım yerin bulunduğum konumdan kaç bayt uzak olduğunu bilmediğim için LJMP(Long JuMP) ile atlama yapıyorum
ORG 0BH	bu satırdan sonraki kodlar 0BH adresinden başlasın bu adres T0(Zamanlayıcı 0) kesmesinin çalıştığı adrestir
LJMP MOTOR	MOTOR etiketine atla burada her bir kesmenin çalışacağı kod bloğu 8 bayt ile sınırlıdır(her kodun bayt olarak kapladığı alan farklıdır yani kod 1 satır 1 bayt yer kaplıyor olarak düşünülmesi yanlıştır) T0 kesmesi için yazacağım kod 8 bayttan daha uzun olabilme ihtimalini düşünerek çalışmasını istediğim kodları ayrı bir yerde yazıyorum ve bu kodları yazdığım yerin bulunduğum konumdan kaç bayt uzak olduğunu bilmediğim için LJMP(Long JuMP) ile atlama yapıyorum
ORG 13H	bu satırdan sonraki kodlar 13H adresinden başlasın bu adres INT1(Harici Kesme 1) kesmesinin çalıştığı adrestir
LJMP STOP	STOP etiketine atla burada her bir kesmenin çalışacağı kod bloğu 8 bayt ile sınırlıdır(her kodun bayt olarak kapladığı alan farklıdır yani kod 1 satır 1 bayt yer kaplıyor olarak düşünülmesi yanlıştır) INT1 kesmesi için yazacağım kod 8 bayttan daha uzun olabilme ihtimalini düşünerek çalışmasını istediğim kodları ayrı bir yerde yazıyorum ve bu kodları yazdığım yerin bulunduğum konumdan kaç bayt uzak olduğunu bilmediğim için LJMP(Long JuMP) ile atlama yapıyorum
ORG 1BH	bu satırdan sonraki kodlar 1BH adresinden başlasın bu adres T1(Zamanlayıcı 1) kesmesinin çalıştığı adrestir
LJMP IKAZ	IKAZ etiketine atla burada her bir kesmenin çalışacağı kod bloğu 8 bayt ile sınırlıdır(her kodun bayt olarak kapladığı alan farklıdır yani kod 1 satır 1 bayt yer kaplıyor olarak düşünülmesi yanlıştır) T1 kesmesi için yazacağım kod 8 bayttan daha uzun olabilme ihtimalini düşünerek çalışmasını istediğim kodları ayrı bir yerde yazıyorum ve bu kodları yazdığım yerin bulunduğum konumdan kaç bayt uzak olduğunu bilmediğim için LJMP(Long JuMP) ile atlama yapıyorum
ORG 30H	bu satırdan sonraki kodlar 30H adresinden başlasın ana programımın bulunduğu adrestir
MAIN: CLR P0.0	P0.0 pinini sıfırla “başlangıçta sistem durmaktadır” dendiği için motorun çalışmadığı anlaşılıyor bu nedenle motorun bağlı olduğu P0.0 pinini lojik 0 yapıyorum
CLR P0.1	P0.1 pinini sıfırla “başlangıçta sistem durmaktadır” dendiği için ikaz lambasının çalışmadığı anlaşılıyor bu nedenle ikaz lambasının bağlı olduğu P0.1 pinini lojik 0 yapıyorum
MOV IE, #8FH	IE kaydedicisine 8FH bilgisini yaz burada 8FH(1000 1111B) yazılmasının nedenine bakarsak 8 bitin yüksek kısmındaki ilk bit EA bitidir ve kesme kullanılacaksa 1 yapılmalıdır 8 bitin yüksek kısmında bizi ilgilendiren başka bit bulunmadığı için kalanına 0 yazıyoruz 8 bitin düşük kısmındaki ilk bit ET1 bitidir T1 kesmesini kullanmamız gerektiği için 1 yazıyoruz ikinci bit EX1 bitidir INT1 kesmesini kullanmamız gerektiği için 1 yazıyoruz üçüncü bit ET0 bitidir T0 kesmesini kullanmamız gerektiği için 1 yazıyoruz son bit EX0 bitidir INT0 kesmesini kullanmamız gerektiği için 1 yazıyoruz
MOV TMOD, #21H	TMOD kaydedicisine 21H bilgisini yaz burada 21H(0010 0001B) yazılmasının nedenine bakarsak

	8 bitin yüksek kısmı T1 ve düşük kısmı T0 içindir her biri için kullanılan 4 biti kısaca şu şekilde açıklayabilirim GATE C/T M1 M0 GATE: zamanlayıcı/sayıcının aktifliğini kontrol eder zamanlayıcı/sayıcının aktif olabilmesi için değeri 0 olmalıdır C/T: zamanlayıcı/sayıcının zamanlayıcı mı sayıcı mı olarak kullanılacağını kontrol eder 0 değeri verilirse zamanlayıcı 1 değeri verilirse sayıcı olur M1 M0: zamanlayıcı/sayıcının modunu belirtir açıklamaya yüksek kısımdan başlarsam T1 zamanlayıcı/sayıcısını kullanacağım için GATE biti 0 olmalıdır T1 zamanlayıcı/sayıcısını zamanlayıcı olarak kullanacağım için C/T biti 0 olmalıdır T1 zamanlayıcı/sayıcısını 50µs sayma amacıyla kullanacağım için sadece MOD 2'i kullanabilirim bu nedenle M1 M0 bitleri 10 olmalıdır düşük kısmı açıklarsam T0 zamanlayıcı/sayıcısını kullanacağım için GATE biti 0 olmalıdır T0 zamanlayıcı/sayıcısını zamanlayıcı olarak kullanacağım için C/T biti 0 olmalıdır T0 zamanlayıcı/sayıcısını 50ms(50 000µs) sayma amacıyla kullanacağım için sadece MOD 1'i kullanabilirim bu nedenle M1 M0 bitleri 01 olmalıdır
MOV TH1, #(-50)	T1 zamanlayıcısının yüksek kısmına -50 bilgisini yaz 50µs sayma yapılacağı için -50 bilgisini yazdım 206 (256-50)da yazabilirdim
MOV TL1, #(-50)	T1 zamanlayıcısının düşük kısmına -50 bilgisini yaz 50µs sayma yapılacağı için -50 bilgisini yazdım 206 (256-50)da yazabilirdim
SJMP $	sonsuz döngü
START: MOV TH0, #HIGH(-50000)	T0 zamanlayıcısının yüksek kısmına -50000 sayısının yüksek kısmını yaz 50ms(50 000µs) sayma yapılacağı için -50000 bilgisini yazdım 15536 (65536-50000)da yazabilirdim
MOV TL0, #LOW(-50000)	T0 zamanlayıcısının düşük kısmına -50000 sayısının düşük kısmını yaz 50ms(50 000µs) sayma yapılacağı için -50000 bilgisini yazdım 15536 (65536-50000)da yazabilirdim
SETB TR0	T0 zamanlayıcısını başlat
SETB TR1	T1 zamanlayıcısını başlat
RETI	kesmeden geri dön
IKAZ: CPL P0.1	P0.1 pinine bağlı olan ikaz lambasını tersle yanıyorsa sönecek sönmüşse yanacak
RETI	kesmeden geri dön
STOP: MOV TH0, #HIGH(-50000)	T0 zamanlayıcısının yüksek kısmına -50000 sayısının yüksek kısmını yaz 50ms(50 000µs) sayma yapılacağı için -50000 bilgisini yazdım 15536 (65536-50000)da yazabilirdim
MOV TL0, #LOW(-50000)	T0 zamanlayıcısının düşük kısmına -50000 sayısının düşük kısmını yaz 50ms(50 000µs) sayma yapılacağı için -50000 bilgisini yazdım 15536 (65536-50000)da yazabilirdim
SETB TR0	T0 zamanlayıcısını başlat
SETB TR1	T1 zamanlayıcısını başlat
RETI	kesmeden geri dön
MOTOR: CPL P0.0	P0.0 pinine bağlı olan motoru tersle çalışıyorsa duracak duruyorsa çalışacak
CLR TR0	T0 zamanlayıcısını durdur
CLR TR1	T1 zamanlayıcısını durdur
JB P0.0, IKAZ_YAK	P0.0 pinindeki bit değeri lojik 1 ise IKAZ_YAK etiketine atla P0.0 pininde motor bağlı olduğu için eğer bu satır start butonuna basıldıktan sonra çalışıyorsa CPL komutu P0.0 pinini terslerken motoru çalıştırır ve bu nedenle ikaz lambasının sürekli yanması gerekir eğer bu satır stop butonuna basıldıktan sonra çalışıyorsa CPL komutu P0.0 pinini terslerken motoru durdurur ve bu nedenle ikaz lambasının sönmesi gerekir
CLR P0.1	P0.1 pinine bağlı olan ikaz lambasını sıfırla ikaz lambasını söndür
LCALL SicaklikDegerlendir	SicaklikDegerlendir altprogramını çağır altprogramın kaç bayt uzaklıkta olduğunu bilmediğim için LCALL kullandım

RETI	kesmeden geri dön
IKAZ_YAK: SETB P0.1	P0.1 pinine bağlı olan ikaz lambasını lojik 1 yap ikaz lambasını yak
RETI	kesmeden geri dön
END	kodu bitir

Burada soruda ikaz lambasının bağlı olduğu pini göremediğim için motorun bağlı olduğu pinden sonraki pindir diye düşündüm ve bu nedenle ikaz lambası için P0.1 pinini kullandım.

İkaz lambasının T1 ile 100 µs periyotla yanıp sönmesi istenmiş. Bu kısım ile ilgili olarak lambanın yanıp söndükten sonra tekrar yanması için geçmesi gereken zaman verilmiş. Yani bana değişim için geçecek zaman gerekli olduğundan dolayı bu verilen zamanın yarısını alarak 50 µs değerinden bir atama yaptım.

“bu proses devam ederken her program çevriminde Soru-2’deki altprogram çağırılacaktır” ifadesinden ben her başlangıç durumuna dönüldüğünde altprogram çağırılacaktır diye düşündüm. Bu nedenle sistemin tekrar durdurulduğu yerde çağırılmasını sağladım.