MİKROİŞLEMCİLER FİNAL PROJESİ: BASİT HESAP MAKİNESİ

SAMİ EREN TÜYSÜZ-221401068

YUNUS EMRE YAŞAR-211401013

ELİF KALENDER-211401008

YAREN KÖSE-211401017

EBRU ŞENER-211401066

MAYIS 2024

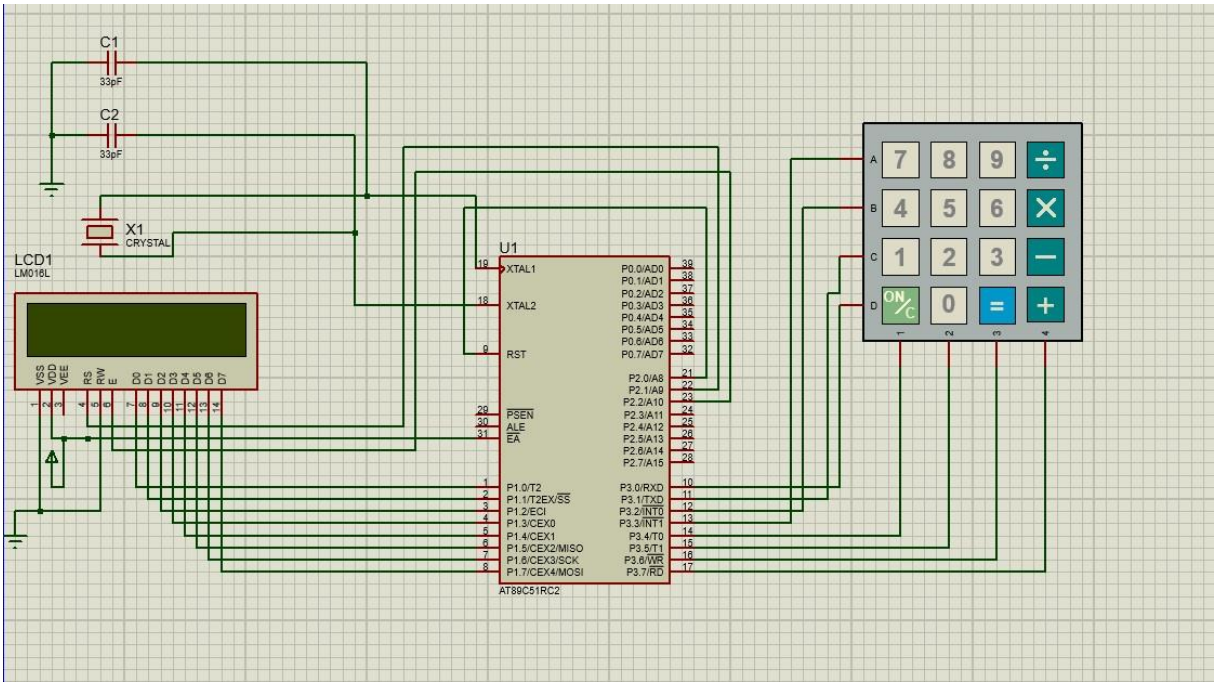
BASİT HESAP MAKİNASI

❖ GİRİŞ

Bu rapor, AT89C51RC2 mikrodenetleyici tabanlı basit bir hesap makinası tasarımını ve uygulanmasını detaylandırmaktadır. Tasarım, 8051 mimarisini temel alarak yapılmıştır ve sadece 8 bitlik hesaplamaları desteklemektedir. Hesap makinası, temel ikili işlemleri gerçekleştirebilmekte olup, 256'dan büyük sayıları işleyemez. Tasarım sürecinde, bir adet LCD ekran ve bir adet keypad kullanılmıştır.

Hesap makinasının ana bileşenleri ve mimarisi incelendikten sonra, kullanılan donanım ve yazılımın detayları açıklanacak ve sonuçlar tartışılacaktır. Bu rapor, basit bir hesap makinası tasarlama sürecinin ayrıntılı bir incelemesini sunmayı amaçlamaktadır.

❖ DEVRE TASARIMI



- **DEVRE ELEMANLARI VE İŞLEVLERİ**

LCD Ekran: LCD ekran, mikrodenetleyicinin verilerini görsel olarak kullanıcıya sunmak için kullanılır. Bu özel uygulamada, sonuçları ve girdileri göstermek için kullanılır.

Keypad (Tuş Takımı): Keypad, kullanıcının sayıları ve işlemleri girmesini sağlar. Tuş takımı, kullanıcının basılan tuşları okuyarak mikrodenetleyiciye sayılar veya işlemler olarak bilgi sağlar.

AT89C51RC2 Mikrodenetleyici: Bu, hesap makinesinin beyin kısmıdır. Programı çalıştırır, tuş girişlerini algılar, işlemleri yapar, sonuçları LCD ekrana gönderir ve LCD ekranın görüntüsünü kontrol eder.

Crystal: Crystal, mikrodenetleyicinin zamanlama işlevi için kullanılır. Mikrodenetleyicinin işlem hızını belirler ve doğru zamanlama sağlar.

33pF Kapasitörler: Crystal ile birlikte kullanılarak, crystalin frekansını sabitlemek ve stabilize etmek için kullanılır.

ASSEMBLY KODLAMA

Assembly kodun başlangıç kısmı, 8-bitlik bir hesap makinesinin temel ayarlarını içermektedir. Bellekteki başlangıç adresini belirlemek üzere "ORG 00H" kullanılmıştır ve hesaplama işlemleri için gerekli olan kayıtlar (R0, R1, R2, R3) ile portlar (P1, P2, P3) tanımlanmıştır.

	KOD	AÇIKLAMA
BAŞLANGIÇ	ORG 00H ; 8 bitli hesap makinesi ; R0 LCD'de komutlar ve veriler için kullanılır ; R1 ilk sayıyı saklar ; R2 işlemi saklar ; R3 ikinci sayıyı saklar ; Port P3 - klavyeden giriş ; Port P2 - sıfırlama + LCD'nin 2 pini (E VE RS)) ; Port P1 - LCD için çıkış YENISAYI EQU P2.7 ; P2.7, ikinci sayıyı yazarken yüksek YENIRAKAM EQU P2.6 ; P2.6, bir sayının ilk rakamını yazarken düşüktür RS EQU P2.1 ; LCD'de RS RW EQU P2.2 ; LCD'de RW	

Bu kod parçacığı, bir LCD ekranı kullanarak bilgi göstermek için gerekli olan başlangıç adımlarını içerir. İlk olarak, LCD'nin başlatılması ve belirli komutların gönderilmesi için hazırlık yapılır. Ardından, ekran açılır, imleç ilk satıra konur ve ekran temizlenir. Sonrasında, LCD kontrol ve veri portları hazırlanır. R4 kaydı sıfırlanır, LCD kontrol portu (muhtemelen P2) sıfırlanır ve LCD veri portu (muhtemelen P3) belirli bir konuma ayarlanır. R1 ve R3 kayıtları da sıfırlanır. Son olarak, ekrana bir karakter göndermek için R2 kaydına '+' atanır. Bu kod, LCD ekranı üzerinde belirli bir karakteri görüntülemek için gerekli olan temel adımları içerir ve LCD'nin doğru şekilde başlatılmasını sağlar.

	KOD	AÇIKLAMA
LCD EKRAN HAZIRLIĞI		

```
; LCD'yi başlatmak ve belirli komutları göndermek için hazırlık yapılır
MOV R0, #38H        ; LCD'yi başlatmak ve 5x7 matris kullanmak için komut
ACALL KOMUT         ; LCD'de komutu yürütmek için alt programı çağır
MOV R0, #0EH        ; Ekranı açmak için komut
ACALL KOMUT
MOV R0, #80H        ; İlk satıra imleç koymak için komut
ACALL KOMUT
MOV R0, #01H        ; Ekranı temizlemek için komut
ACALL KOMUT
; LCD kontrol ve veri portlarını hazırlar
MOV R4, #00H        ; R4'ü sıfırla
MOV P2, #00H        ; P2'yi sıfırla (muhtemelen LCD kontrol portu)
MOV P3, #0FEH       ; L1 yerinde LCD'yi başlat (1. satır aktif) (muhtemelen LCD veri portu)
MOV R3, #00H        ; R3'ü sıfırla
MOV R1, #00H        ; R1'i sıfırla
MOV R2, #'+'        ; Belirli bir karakter (muhtemelen artı işareti) için bir değer atanır
```

Bu kod, bir dizi P3 portu pinini kontrol ederek LCD ekranın satırlarını tarar. Her bir JNB (Jump if Bit Not Set) komutu, belirli bir pinin durumunu kontrol eder. Eğer pin durumu 0 ise (yani, LCD ekranın o satırında bir sinyal varsa), ilgili etikete (C1, C2, C3, C4) atlama yapar. Eğer pin durumu 1 ise (yani, LCD ekranın o satırında bir sinyal yoksa), atlama yapmaz ve sonraki satıra geçer. Bu kod, bir dizi P3 portu pinini kontrol ederek LCD ekranın satırlarını tarar. Her bir JNB (Jump if Bit Not Set) komutu, belirli bir pinin durumunu kontrol eder. Eğer pin durumu 0 ise (yani, LCD ekranın o satırında bir sinyal varsa), ilgili etikete (C1, C2, C3, C4) atlama yapar. Eğer pin durumu 1 ise (yani, LCD ekranın o satırında bir sinyal yoksa), atlama yapmaz ve sonraki satıra geçer.

	KOD	AÇIKLAMA
L1 ETİKETİ	L1: JNB P3.0, C1 ; 0 ise C1'e atlar (her zaman 0'da başlar) JNB P3.1, C2 ; 0 ise C2'e atlar JNB P3.2, C3 ; 0 ise C3'e atlar JNB P3.3, C4 ; 0 ise C4'e atlar SJMP L1 ; satır kontrollerine dönmek için	

C1 etiketi,LCD ekranı üzerinde belirli işlevlere erişimi sağlayan bir düğme kontrol mekanizmasını yönetir. Her bir düğme, belirli bir işleve atanmıştır. Örneğin, "ON" düğmesi belirli bir işlemi başlatırken, "+" düğmesi farklı bir işlemi tetikler. Kod, her bir düğmenin durumunu kontrol eder ve belirli bir düğmeye basıldığında ilgili işlemleri gerçekleştirir. Daha sonra, LCD ekranının belirli satırları etkinleştirilir veya devre dışı bırakılır, böylece kullanıcıya belirli işlevlerin görüntülenmesi sağlanır. Kod, düğme durumlarını sürekli olarak kontrol eder ve LCD ekranının durumunu günceller, böylece kullanıcının etkileşimlerine anında yanıt verebilir. Bu kod, kullanıcı arayüzünün işlevselliğini sağlamak için önemli bir temel oluşturur.

	KOD	AÇIKLAMA
C1 ETİKETİ	C1: JNB P3.4, ATLA_DUGME_ON ; ON düğmesine basılırsa atlar JNB P3.5, ATLA_DUGME_SIFIR ; '0' düğmesine basılırsa atlar JNB P3.6, ATLA_DUGME_ESIT ; '=' düğmesine basılırsa atlar JNB P3.7, ATLA_DUGME_ARTI ; '+' düğmesine basılırsa atlar SETB P3.0 ; 1. satırı deaktif et CLR P3.1 ; 2. satırı aktif et SJMP L1 ; satır kontrollerine dönmek için	

C2, C3 VE C4 etiketleri C1 etiketiyle aynı amacı taşır, hepsi birlikte LCD ekranı üzerindeki belirli işlevlere erişimi sağlayan bir düğme kontrol mekanizmasını yönetir. Daha önce belirtildiği gibi her bir düğme, belirli bir işleve atanmıştır.

	KOD	AÇIKLAMA
C2 ETİKETİ	C2: JNB P3.4, ATLA_DUGME_1 ; '1' düğmesine basılırsa atlar JNB P3.5, ATLA_DUGME_2 ; '2' düğmesine basılırsa atlar JNB P3.6, ATLA_DUGME_3 ; '3' düğmesine basılırsa atlar JNB P3.7, ATLA_DUGME_EKSI ; '-' düğmesine basılırsa atlar SETB P3.1 ; 2. satırı deaktif et CLR P3.2 ; 3. satırı aktif et SJMP L1 ; satır kontrollerine dönmek için	

	KOD	AÇIKLAMA
C3 ETİKETİ	C3: JNB P3.4, ATLA_DUGME_4 ; '4' düğmesine basılırsa atlar JNB P3.5, ATLA_DUGME_5 ; '5' düğmesine basılırsa atlar JNB P3.6, ATLA_DUGME_6 ; '6' düğmesine basılırsa atlar JNB P3.7, ATLA_DUGME_CARPI ; 'x' düğmesine basılırsa atlar SETB P3.2 ; 3. satırı deaktif et CLR P3.3 ; 4. satırı aktif et SJMP L1 ; satır kontrollerine dönmek için	

	KOD	AÇIKLAMA
C4 ETİKETİ	C4: JNB P3.4, ATLA_DUGME_7 ; '7' düğmesine basılırsa atlar JNB P3.5, ATLA_DUGME_8 ; '8' düğmesine basılırsa atlar JNB P3.6, ATLA_DUGME_69 ; '9' düğmesine basılırsa atlar JNB P3.7, ATLA_DUGME_BOLME ; '/' düğmesine basılırsa atlar SETB P3.2 CLR P3.3 SJMP L1	

Bu kod bloğu, belirli düğmelerin basılması durumunda ilgili işlevlere atlama yapmak için kullanılır. Her bir düğme, belirli bir işlevi temsil eder. Örneğin, "ON" düğmesi belirli bir işlemi başlatırken, "+" düğmesi farklı bir işlemi tetikler. Kod, belirli bir düğme basıldığında ilgili işlevin gerçekleştirileceği etiketlere atlama yapar.

Örneğin, ATLA_DUGME_ON etiketi, "ON" düğmesine basıldığında LJMP komutuyla DUGME_ON etiketine atlama yapar. Benzer şekilde, diğer düğmeler için de ilgili etiketlere atlama yapılır.

Bu kod bloğu, düğme basıldığında hangi işlemin gerçekleştirileceğini belirler ve ilgili işlevin bulunduğu etikete atlama yaparak programın o kısmına yönlendirir. Bu sayede, kullanıcının düğmeler aracılığıyla belirli işlevleri etkinleştirmesi sağlanır.

	KOD	AÇIKLAMA
ASSEMBLY KOD	ATLA_DUGME_ON: LJMP DUGME_ON ;DUGME_ON etiketine atla ATLA_DUGME_SIFIR: LJMP DUGME_ZERO ;DUGME_ZERO etiketine atla ATLA_DUGME_1: LJMP DUGME_1 ;DUGME_1 etiketine atla ATLA_DUGME_2: LJMP DUGME_2 ;DUGME_2 etiketine atla ATLA_DUGME_3: LJMP DUGME_3 ;DUGME_3 etiketine atla ATLA_DUGME_4: LJMP DUGME_4 ;DUGME_4 etiketine atla ATLA_DUGME_5: LJMP DUGME_5 ;DUGME_5 etiketine atla ATLA_DUGME_6: LJMP DUGME_6 ;DUGME_6 etiketine atla ATLA_DUGME_7: LJMP DUGME_7 ;DUGME_7 etiketine atla ATLA_DUGME_8: LJMP DUGME_8 ;DUGME_8 etiketine atla ATLA_DUGME_9: LJMP DUGME_9 ;DUGME_9 etiketine atla ATLA_DUGME_ARTI: LJMP DUGME_ARTI ;DUGME_ARTI etiketine atla ATLA_DUGME_EKSI: LJMP DUGME_EKSI ;DUGME_EKSI etiketine atla ATLA_DUGME_CARPI: LJMP DUGME_CARPI ;DUGME_CARPI etiketine atla ATLA_DUGME_BOLME: LJMP DUGME_BOLME ;DUGME_BOLME etiketine atla ATLA_DUGME_ESIT: LJMP DUGME_ESIT ;DUGME_ESIT etiketine atla	

DUGME_ON etiketi, "ON" düğmesine basıldığında gerçekleştirilecek işlevi belirtir. İlk olarak, P2 portunun 0. pinine yüksek seviyeli bir sinyal uygulanarak bir reset işlemi tetiklenir. Ardından, programın belirli bir noktadan devam etmesi için L1 etiketine atlama yapılır. Bu sayede, "ON" düğmesine basıldığında belirli bir cihazın veya uygulamanın başlangıç durumuna dönmesi sağlanır. Bu kod, kullanıcıya cihazı yeniden başlatma veya sıfırlama olanağı sunar.

	KOD	AÇIKLAMA
DUGME_ON	DUGME_ON: SETB P2.0 ; reset'i aktifleştir LJMP L1	

DUGME_SIFIR etiketi, "0" düğmesine basıldığında gerçekleştirilecek işlevi belirler. İlk olarak, '0' karakteri R0 kaydına taşınır ve ardından bu karakter bir alt program olan SAYI'da saklanır. Daha sonra, YAZDIR adlı başka bir alt program aracılığıyla karakter LCD ekranında görüntülenir. Son olarak, program L1 etiketine atlama yaparak devam eder. Bu kod bloğu, kullanıcının "0" düğmesine basarak girdiği sayıları LCD ekranında görüntülemesine olanak tanır, bu da kullanıcının girdiği bilgileri doğrulamasına veya ilgili işlemleri gerçekleştirmesine olanak sağlar.

	KOD	AÇIKLAMA
DUGME_SIFIR	DUGME_SIFIR: MOV R0, #'0' ; '0' karakterini R0'a taşı ACALL SAYI ; numarayı sakla ACALL YAZDIR ; karakteri LCD'de yazdır LJMP L1	

	KOD	AÇIKLAMA
DUGME_ESIT DUGME_ARTI DUGME_1,2,3	DUGME_ESIT: MOV R0, #'=' ; '=' KARAKTERINI R0'E TAŞI ACALL YAZDIR ; EKRANA YAZDIR ACALL SONUC ; İŞLEMİ GERÇEKLEŞTİR LJMP L1 DUGME _ARTI: MOV R0, #'+' ; '+' KARAKTERINI R0'E TAŞI ACALL ISLEM ; İŞLEMİ R2'DE SAKLA ACALL YAZDIR ; EKRANA YAZDIR LJMP L1 DUGME _1: MOV R0, #'1' ; '1' KARAKTERINI R0'E TAŞI ACALL SAYI ; SAYIYI SAKLA ACALL YAZDIR ; EKRANA YAZDIR LJMP L1 DUGME _2: MOV R0, #'2' ; '2' KARAKTERINI R0'E TAŞI ACALL SAYI ; SAYIYI SAKLA ACALL YAZDIR ; EKRANA YAZDIR LJMP L1 ACALL SAYI ; SAYIYI SAKLA ACALL YAZDIR ; EKRANA YAZDIR LJMP L1	

Bu kod içerisindeki etiketler belirli işlevlere sahip alt rutinleri temsil eder. " DUGME _ESIT" etiketi, eşittir işaretiyle ilişkilendirilmiş bir alt rutini, eşittir işaretini belirli bir register'a yükleyip ekrana yazdırmak ve sonucu hesaplamak için ilgili alt rutinleri çağırmak için kullanılır. " DUGME _ARTI" etiketi, toplama işaretiyle ilişkilendirilmiş bir alt rutini temsil eder ve toplama işaretini belirli bir register'a yükleyip işlemi gerçekleştirmek için ilgili alt rutinleri çağırmak için kullanılır. " DUGME _1" ve " DUGME _2" etiketleri ise sırasıyla 1 ve 2 rakamlarıyla ilişkilendirilmiş alt rutinleri temsil eder; belirli bir rakamı belirli bir register'a yükleyip ekrana yazdırmak ve gerektiğinde işlem yapmak için ilgili alt rutinleri çağırmak için kullanılır. Bu etiketler, kodu parçalara böler ve her bir parçanın ne işe yaradığını açıklar, kodun anlaşılmasını kolaylaştırır.

	KOD	AÇIKLAMA
DUGME_DAHA_AZ		

```
DUGME _3:       MOV R0, #'3'      ; '3' KARAKTERINI R0'E TAŞI
                ACALL SAYI        ; SAYIYI SAKLA
                ACALL YAZDIR      ; EKRANA YAZDIR
                LJMP L1
DUGME _DAHA_AZ: MOV R0, #'-'      ; '-' KARAKTERINI R0'E TAŞI
                ACALL ISLEM       ; İŞLEMİ R2'DE SAKLA
                ACALL YAZDIR      ; EKRANA YAZDIR
                LJMP L1

DUGME _4:       MOV R0, #'4'      ; '4' KARAKTERINI R0'E TAŞI
                ACALL SAYI        ; SAYIYI SAKLA
                ACALL YAZDIR      ; EKRANA YAZDIR
                LJMP L1

DUGME _5:       MOV R0, #'5'      ; '5' KARAKTERINI R0'E TAŞI
                ACALL SAYI        ; SAYIYI SAKLA
                ACALL YAZDIR      ; EKRANA YAZDIR
                LJMP L1
```

Bu kod bloğu, farklı rakamları ve çıkarma işaretini işleyen birkaç alt rutini içeriyor. " DUGME _3" etiketi, 3 rakamını işlerken, " DUGME _ DAHA_AZ " çıkarma işaretini işler. " DUGME _4" ve " DUGME _5" etiketleri ise sırasıyla 4 ve 5 rakamlarını işler. Her etiket, belirli bir rakam veya işareti belirli bir register'a yükleyip ekrana yazdırmak ve gerektiğinde işlem yapmak için ilgili alt rutinleri çağırır. Bu blok, kodun işlevselliğini genişletir ve okunabilirliğini artırır.

	KOD	AÇIKLAMA
DUGME_KEZ DUGME_ BOLME		

```
DUGME _6:       MOV R0, #'6'      ; '6' KARAKTERINI R0'E TAŞI
                ACALL SAYI        ; SAYIYI SAKLA
                ACALL DISPLAY     ; EKRANA YAZDIR
                LJMP L1
DUGME _KEZ:     MOV R0, #''       ; '' KARAKTERINI R0'E TAŞI
                ACALL ISLEM       ; İŞLEMİ R2'DE SAKLA
                ACALL YAZDIR      ; EKRANA YAZDIR
                LJMP L1
DUGME _7:       MOV R0, #'7'      ; '7' KARAKTERINI R0'E TAŞI
                ACALL SAYI        ; SAYIYI SAKLA
                ACALL YAZIDR      ; EKRANA YAZDIR
                LJMP L1
DUGME _8:       MOV R0, #'8'      ; '8' KARAKTERINI R0'E TAŞI
                ACALL SAYI        ; SAYIYI SAKLA
                ACALL YAZIDR      ; EKRANA YAZDIR
                LJMP L1
DUGME _9:       MOV R0, #'9'      ; '9' KARAKTERINI R0'E TAŞI
                ACALL SAYI        ; SAYIYI SAKLA
                ACALL YAZDIR      ; EKRANA YAZDIR
               LJMP L1
DUGME _ BOLME:  MOV R0, #'/'      ; '/' KARAKTERINI R0'E TAŞI
                ACALL ISLEM       ; İŞLEMİ R2'DE SAKLA
                ACALL YAZDIR      ; EKRANA YAZDIR
                LJMP L1
```

Bu kod bloğu, çeşitli rakamların yanı sıra çarpma ve bölme işaretlerini işleyen bir dizi alt rutini içeriyor. " DUGME _6" etiketi 6 rakamını, " DUGME _7", " DUGME _8" ve " DUGME _9" etiketleri sırasıyla 7, 8 ve 9 rakamlarını işlerken, " DUGME _KEZ" çarpma işaretini ve " DUGME _ BOLME " bölme işaretini işler. Her bir etiket, belirli bir rakam veya işareti belirli bir register'a yükleyip ekrana yazdırmak ve gerektiğinde işlem yapmak için ilgili alt rutinleri çağırır. Bu blok, kodun işlevselliğini daha da artırır ve okunabilirliğini artırır.elde edilir ve bu sayı üzerinde işlemler yapılabilir.

	KOD	AÇIKLAMA
YAZDIR	YAZDIR: MOV P1, R0 ; KARAKTERI ÇIKIŞA (P1) TAŞI SETB RS ; RS'Yİ (REGISTER SELECT) VERİ MODUNA AYARLA SETB RW ; LCD'DE OKUMA/YAZMA İZİNİNİ SERBEST BIRAK (YÜKSEK SEVİYE) CLR RW ACALL DELAY ; 0.25s GEÇİKME ÇAĞIR (BİRDEN FAZLA KEZ YAZDIRMAYI ÖNLEMEK İÇİN) RET ; Alt programdan dön	

Bu kod parçası, bir hesap makinesi programında kullanılabilir ve kullanıcıdan alınan işlem operatörünü saklar. YENI_SAYI bayrağını set ederek, bir sonraki rakamın ikinci sayıya ait olduğunu belirtir. ILK_HANE_KAYDEDILDI bayrağını sıfırlayarak, bir sonraki rakamın ilk sayıya ait olduğunu belirtir. Son olarak, işlem operatörünü R2 kaydında saklar ve alt programdan döner. Bu şekilde, kullanıcı tarafından belirlenen işlem operatörü program tarafından saklanarak, hesaplama yapılacak işlem belirlenmiş olur.

	KOD	AÇIKLAMA
KOMUT	KOMUT: MOV P1, R0 ; KOMUTU ÇIKIŞA TAŞI - LCD GİRİŞİ CLR RS ; RS'Yİ (REGISTER SELECT) KOMUT MODUNA AYARLA SETB RW ; LCD'DE OKUMA/YAZMA İZİNİNİ SERBEST BIRAK (YÜKSEK SEVİYE) CLR RW ACALL DELAY ; 0.25s GEÇİKME ÇAĞIR RET	

Bu kod bloğu LCD ekranın kontrol komutlarını işlemek için gereken adımları içerir. İlk olarak, R0 register'ındaki komut P1 çıkış portuna taşınır. RS pinini temizleyerek LCD ekranın komut alım moduna geçiş yapılır. RW pinini yüksek seviyeye ayarlayarak LCD ekranın okuma/yazma iznini serbest bırakılır, böylece mikrodenetleyiciye LCD ekran üzerinde komut gönderme izni verilir. Ardından bir gecikme çağrılarak işlem sırasında beklenir ve son olarak alt programdan dönülerek COMMAND alt rutini sonlandırılır ve ana programa geri dönülür. Bu blok, mikrodenetleyicinin LCD ekran üzerinde kontrol komutlarını işlemesini sağlar.

	KOD	AÇIKLAMA

SAYI	SAYI:
	JB YENI_SAYI, IKINCI_NUMARA ; EĞER İKİNCİ SAYIYSA ATLAR
	JB ILK_HANE_KAYDEDILDI, YENI_RAKAM_ISLEM ; EĞER İLK SAYININ İLK HANESİ DEĞİLSE ATLAR
	MOV A, R0 ; KARAKTERİ AKÜMÜLATÖRE TAŞIR
	SUBB A, #30H ; BU KARAKTERİ SAYIYA DÖNÜŞTÜRÜR
	MOV R1, A ; BU SAYIYI R1'E KAYDEDER
	SETB ILK_HANE_KAYDEDILDI ; ILK_HANE_KAYDEDILDI PININI AYARLA => İLK SAYININ İLK HANESİ KAYDEDİLDİ
	RET	

Bu kod bloğu, bir karakterin sayıya dönüştürülmesini ve sayının saklanmasını sağlar. İlk olarak, kod, YENI_SAYI ve IKINCI_NUMARA bayraklarını kontrol eder. Eğer IKINCI_NUMARA bayrağı ayarlanmışsa, yani ikinci sayının karakteri ise bu bloğu atlar. Eğer ILK_HANE_KAYDEDILDI bayrağı ayarlanmamışsa, yani ilk sayının ilk hanesi değilse de atlar. Karakter, akümülatöre (A) taşınır ve 30H değeri çıkarılarak karakter sayıya dönüştürülür. Elde edilen sayı, R1 register'ına kaydedilir ve ILK_HANE_KAYDEDILDI bayrağı ayarlanarak ilk sayının ilk hanesinin kaydedildiği işaretlenir. Son olarak, blok alt programdan döner. Bu kod bloğu, karakterin sayıya dönüştürülüp saklanması ve işaretlenmesi işlemlerini gerçekleştirir.

	KOD	AÇIKLAMA
ASSEMBLY KOD		

```
YENI_RAKAM_ISLEM:       ; Alt program başlangıcı
    MOV A, R0     ; R0'daki değeri A'ya kopyala
    MOV B, #10D     ; B'ye 10 sayısını yükle (çarpmak için)
    SUBB A, #30H    ; '0' karakterinin ASCII değerini çıkararak rakamın değerini elde  et
    MOV R7, A      ; Rakamın değerini R7'ye kopyala
    MOV A, R1      ; R1'deki değeri A'ya kopyala
    MUL AB        ; A ve B'yi çarpar, sonucu A ve B'ye yazar (çarpım sonucunu elde et)
    MOV R6, B      ; Çarpım sonucunu R6'ya kopyala
    CJNE R6, #00H, ATLA_ TASMA   ; Eğer çarpım sonucu sıfırdan farklı ise ATLA_ TASMA 'ya atla
    ADD A, R7      ; Çarpım sonucu sıfırsa, rakamın değerini toplama ekle
    JC ATLA_ TASMA    ; Eğer taşma olduysa, ATLA_ TASMA ya atla
    MOV R1, A      ; Sonucu R1'e kopyala
    SETB ILK_HANE_KAYDEDILDI    ; ILK_HANE_KAYDEDILDI bayrağını ayarla
    RET          ; Alt programdan dön
```

Bu kod bir basit hesaplama makinesinde kullanılabilecek bir alt programı temsil ediyor. Program, R0 ve R1 kayıtlarında saklanan iki basamaklı bir sayının çarpımını hesaplar. İlk olarak, her bir rakamın ASCII değeri '0' karakterinin ASCII değerinden çıkarılarak gerçek değerleri elde edilir. Ardından, ikinci rakam 10 ile çarpılarak ikinci rakamın değeri bulunur. Çarpma işlemi sıfır olana kadar devam eder ve her adımda çarpım sonucuna ilk rakamın değeri eklenir. Eğer çarpım sonucu 9'dan büyükse veya çarpma işlemi sırasında taşma olursa, ILK_HANE_KAYDEDILDI bayrağı ayarlanır. Son olarak, çarpımın sonucu R1 kaydında saklanır. Bu şekilde, basit bir çarpma işlemi gerçekleştirilir ve taşma durumunda kullanıcıya uyarı verilir.

	KOD	AÇIKLAMA
ASSEMBLY KOD		

```
IKINCI_NUMARA:          ; İkinci rakamın alınması için alt program başlangıcı
    JB ILK_HANE_KAYDEDILDI, YENI_RAKAM_ISLEM2 ; Eğer ILK_HANE_KAYDEDILDI bayrağı set edilmişse (yani ilk rakamın işlenmesi tamamlandıysa),YENI_RAKAM_ISLEM2 etiketine atla
    MOV A, R0           ; A'ya R0 kaydındaki değeri kopyala (ikinci rakamın ASCII değeri)
    SUBB A, #30H        ; '0' karakterinin ASCII değerini çıkararak, rakamın değerini elde et
    MOV R3, A           ; Rakamın değerini R3 kaydında sakla
    SETB ILK_HANE_KAYDEDILDI        ; ILK_HANE_KAYDEDILDI bayrağını set et (ilk rakam işlendi)
    RET                 ; Alt programdan dön
```

Bu kod parçası, ikinci rakamın alınması için bir alt programı temsil ediyor. Kodun çalışma mantığı şu şekilde işliyor: İlk olarak, ILK_HANE_KAYDEDILDI bayrağının durumuna bakılarak (eğer set edilmişse) işlem devam ediyor. Eğer bayrak set edilmişse, yani ilk rakamın işlenmesi tamamlanmışsa, ikinci rakamın alınması için gerekli adımlar gerçekleştiriliyor.

	KOD	AÇIKLAMA
ASSEMBLY KOD		

```
YENI_RAKAM_ISLEM2:          ; İkinci rakamın işlenmesi için alt program başlangıcı
    MOV A, R0               ; A'ya R0 kaydındaki değeri kopyala (yeni rakamın ASCII değeri)
    MOV B, #10D             ; B'ye 10 sayısını yükle (çarpmak için)
    SUBB A, #30H            ; '0' karakterinin ASCII değerini çıkararak rakamın değerini elde et
    MOV R7, A               ; Rakamın değerini R7 kaydına kopyala
    MOV A, R3               ; A'ya R3 kaydındaki değeri kopyala (ikinci sayının değeri)
    MUL AB                  ; A ve B'yi çarpar, sonucu A ve B'ye yazar (ikinci sayıyı 10 ile çarpar)
    MOV R6, B               ; Çarpımın sonucunun daha anlamlı kısmını R6'ya kopyala
    CJNE R6, #00H, ATLA_ TASMA ; Eğer R6'da bir değer varsa, 8 biti aşmış demektir (taşma kontrolü)
    ADD A, R7               ; Eğer taşma olmamışsa, ikinci sayıyı 10 ile çarparak elde edilen değeri ekler
    JC ATLA_ TASMA          ; Eğer taşma olursa, ATLA_ TASMA etiketine atla
    MOV R3, A               ; Sonucu R3 kaydına kopyala (ikinci sayının yeni değeri)
    SETB ILK_HANE_KAYDEDILDI ; ILK_HANE_KAYDEDILDI bayrağını ayarla (yeni rakam işlendi)
    RET                     ; Alt programdan dön
```

Bu program, bir basit hesap makinesinde kullanılabilir ve ikinci rakamın alınması ve işlenmesini sağlar. İlk rakamın alınması ve işlenmesi için YENI_RAKAM_ISLEM alt programı kullanılırken, ikinci rakamın alınması ve işlenmesi için YENI_RAKAM_ISLEM2 alt programı kullanılır.

Bu alt programlar sayesinde, kullanıcı bir sayı girdiğinde ilk rakam alınır ve işlenir. Daha sonra kullanıcı bir rakam daha girdiğinde, bu rakamın alınması ve ilk rakamla birleştirilmesi sağlanır. Bu şekilde, iki basamaklı bir sayı

	KOD	AÇIKLAMA

ASSEMBLY KOD	ISLEM: ; İşlem belirleme alt programı başlangıcı
	SETB YENI_SAYI ; YENI_SAYI bayrağını set et, bir sonraki rakamın ikinci sayıya ait olduğunu belirtir
	CLR ILK_HANE_KAYDEDILDI ; ILK_HANE_KAYDEDILDI bayrağını sıfırla, bir sonraki rakamın ilk sayıya ait olduğunu belirtir
	MOV A, R0 ; A'ya R0 kaydındaki değeri kopyala (işlem operatörü)
	MOV R2, A ; R2'ye A'daki değeri kopyala (işlem operatörünü sakla)
	RET ; Alt programdan dön	

Bu kod parçası, bir hesap makinesi programında kullanılabilir ve kullanıcıdan alınan işlem operatörünü saklar. YENI_SAYI bayrağını set ederek, bir sonraki rakamın ikinci sayıya ait olduğunu belirtir. ILK_HANE_KAYDEDILDI bayrağını sıfırlayarak, bir sonraki rakamın ilk sayıya ait olduğunu belirtir. Son olarak, işlem operatörünü R2 kaydında saklar ve alt programdan döner. Bu şekilde, kullanıcı tarafından belirlenen işlem operatörü program tarafından saklanarak, hesaplama yapılacak işlem belirlenmiş olur.

	KOD	AÇIKLAMA
ASSEMBLY KOD		

```
TOPLAMA_ISLEMI: ; Sonucu hesapla alt programı başlangıcı
    CJNE R2, #'+', CIKARMA    ; Eğer işlem operatörü '+' ise CIKARMA etiketine atla (toplama işlemi)
    MOV A, R1                 ; A'ya R1 kaydındaki değeri kopyala (ilk sayıyı al)
    CLR C                     ; Carry'yi temizle
    ADD A, R3                 ; İlk sayıya ikinci sayıyı ekle
    JC ATLA_ TASMA            ; Eğer taşma olduysa, ATLA_ TASMA etiketine atla
    MOV R5, #0H               ; R5 kaydına 0 değerini koy (bölme işlemi için gereken ayar)
    MOV R4, A                 ; Sonucu R4 kaydına kopyala
    LJMP EKRANA_YAZDIRMA      ; EKRANA_YAZDIRMA etiketine atla (sonucu yazdır)
```

Bu kod parçası, bir hesap makinesinde toplama işlemini gerçekleştirmek için kullanılabilir. İşlem, R1 ve R3 kayıtlarında saklanan iki sayıyı toplar ve sonucu R4 kaydına kaydeder. İlk olarak, R2 kaydındaki işlem operatörü kontrol edilir. Eğer işlem operatörü '+' ise, ADD A, R3 komutuyla A kaydına R1 ve R3 kayıtlarındaki değerlerin toplamı atanır. Eğer bu toplama işlemi sırasında taşma olursa (JC ATLA_ TASMA), taşma durumunu kontrol etmek üzere belirlenmiş bir etikete (ATLA_ TASMA) atlanır. Taşma olmazsa, sonuç R4 kaydına kopyalanır ve EKRANA_YAZDIRMA etiketine atlanarak TOPLAMA_ISLEMI işlemine geçilir.

	KOD	AÇIKLAMA

ASSEMBLY KOD	CIKARMA: ; Çıkarma işlemi
	CJNE R2, #'-', CARPMA ; Eğer işlem operatörü '-' ise CARPMA etiketine atla (çıkarma işlemi)
	MOV A, R1 ; A'ya R1 kaydındaki değeri kopyala (ilk sayıyı al)
	CLR C ; Carry'yi temizle
	SUBB A, R3 ; İlk sayıdan ikinci sayıyı çıkar
	JC ATLA_ TASMA ; Eğer taşma olduysa, ATLA_ TASMA etiketine atla
	MOV R5, #0H ; R5 kaydına 0 değerini koy (bölme işlemi için gereken ayar)
	MOV R4, A ; Sonucu R4 kaydına kopyala
	LJMP EKRANA_YAZDIRMA ; EKRANA_YAZDIRMA etiketine atla (sonucu yazdır)	

Bu kod parçası, bir hesap makinesindeki çıkarma işlemini gerçekleştirir. İlk olarak, R2 kaydındaki işlem operatörünü (- işareti) kontrol eder. Eğer işlem operatörü çıkarma işaretini temsil ediyorsa (CJNE R2, #'-', CARPMA), R1 kaydındaki ilk sayıyı (MOV A, R1) alır. Daha sonra, CLR C komutuyla taşıma (carry) bayrağını temizler ve ikinci sayı olan R3 kaydındaki değeri (SUBB A, R3) ilk sayıdan çıkarır. Son olarak, eğer çıkarma işlemi sırasında bir taşma (carry) olursa (JC ATLA_ TASMA), program ATLA_ TASMA etiketine atlar. Taşma olmazsa, R5 kaydına 0 değerini koyar (MOV R5, #0H), sonucu R4 kaydına kopyalar (MOV R4, A) ve sonucu yazdırmak için EKRANA_YAZDIRMA etiketine atlar (LJMP EKRANA_YAZDIRMA). Bu şekilde, çıkarma işlemi yapılır ve sonucu doğru bir şekilde işaretlenmiş şekilde saklanır.

	KOD	AÇIKLAMA
ASSEMBLY KOD		

```
CARPMA:          ; Çarpma işlemi
    CJNE R2, #'*', BOLME    ; Eğer işlem operatörü '*' ise BOLME etiketine atla (çarpma işlemi)
    MOV A, R1           ; A'ya R1 kaydındaki değeri kopyala (ilk sayıyı al)
    MOV B, R3           ; B'ye R3 kaydındaki değeri kopyala (ikinci sayıyı al)
    MUL AB              ; A ve B'yi çarpar, sonucu A ve B'ye yazar (çarpma işlemi)
    MOV R7, B           ; Çarpımın sonucunun daha anlamlı kısmını R7'ye kopyala
    CJNE R7, #0H, TASMA     ; Eğer R7'de bir değer varsa, 8 biti aşmış demektir (taşma kontrolü)
    MOV R5, #0H         ; R5 kaydına 0 değerini koy (bölme işlemi için gereken ayar)
    MOV R4, A           ; Sonucu R4 kaydına kopyala
    LJMP EKRANA_YAZDIRMA        ; EKRANA_YAZDIRMA etiketine atla (sonucu yazdır)
```

Bu kod parçası, çarpma işlemi gerçekleştirir. İlk olarak, R2 kaydındaki işlem operatörünü (* işareti) kontrol eder. Eğer işlem operatörü çarpma işaretini temsil ediyorsa (CJNE R2, #'*', BOLME), R1 kaydındaki ilk sayıyı (MOV A, R1) ve R3 kaydındaki ikinci sayıyı (MOV B, R3) alır. Daha sonra, MUL AB komutuyla A ve B'yi çarpar ve sonucu A ve B'ye yazar. Son olarak, eğer çarpma işlemi sırasında bir taşma (carry) olursa (CJNE R7, #0H, TASMA), program TASMA etiketine atlar. Taşma olmazsa, R5 kaydına 0 değerini koyar (MOV R5, #0H), sonucu R4 kaydına kopyalar (MOV R4, A) ve sonucu yazdırmak için EKRANA_YAZDIRMA etiketine atlar (LJMP EKRANA_YAZDIRMA). Bu şekilde, çarpma işlemi yapılır ve sonucu doğru bir şekilde işaretlenmiş şekilde saklanır.

	KOD	AÇIKLAMA

ASSEMBLY KOD	BOLME: ; Bölme işlemi
	MOV A, R1 ; A'ya R1 kaydındaki değeri kopyala (bölüneni al)
	MOV B, R3 ; B'ye R3 kaydındaki değeri kopyala (böleni al)
	DIV AB ; A'yı B'ye böl, sonucu A'ya, kalanı B'ye yaz
	MOV R4, A ; Sonucu R4 kaydına kopyala
	MOV R5, B ; Kalanı R5 kaydına kopyala
	LJMP EKRANA_YAZDIRMA ; EKRANA_YAZDIRMA etiketine atla (sonucu yazdır)
	
	ATLA_ TASMA: ; Taşma durumunda atlanacak yer
	LJMP TASMA ; TASMA etiketine atla (taşma durumunu işlemek için)	

Bu kod parçası, bir bölme işlemi gerçekleştirir. İlk olarak, R1 kaydındaki ilk sayıyı A kaydına kopyalar ve R3 kaydındaki ikinci sayıyı B kaydına kopyalar. Ardından, DIV AB komutuyla A'yı B'ye böler ve bölme işleminin sonucunu A'ya, kalanı ise B'ye yazar. Son olarak, bölme işleminin sonucunu R4 kaydına kopyalar ve kalanı R5 kaydına kopyalar. Sonucu yazdırmak için EKRANA_YAZDIRMA etiketine atlar. Bu şekilde, bölme işlemi yapılır ve sonucu doğru bir şekilde işaretlenmiş şekilde saklanır.

	KOD	AÇIKLAMA

ASSEMBLY KOD	EKRANA_YAZDIRMA: ; Sonucu yazdırma işlemi başlangıcı
	CJNE R3, #0D, NORMAL ; Eğer ikinci sayı 0 değilse, NORMAL etiketine atla
	CJNE R2, #'/', NORMAL ; Eğer işlem operatörü bölme işareti değilse, NORMAL etiketine atla
	MOV R0, #0C0H ; Eğer bölme işlemi sırasında ikinci sayı 0 ise, imleci ikinci satıra taşı
	ACALL COMMAND ; LCD ekranında belirtilen komutu çalıştır
	MOV DPTR, #HATA ; Hata mesajının bulunduğu bellek adresini DPTR kaydına yükle
	CLR C ; Taşma (carry) bayrağını temizle
	MOV R7, #0D ; R7 kaydına 0 değerini koy	

Bu kod parçası, sonucun yazdırılmasını ve özel durumların kontrolünü içerir. İlk olarak, R3 kaydındaki ikinci sayının 0 olup olmadığını kontrol eder. Eğer 0 değilse (CJNE R3, #0D, NORMAL), veya işlem operatörü bölme işareti değilse (CJNE R2, #'/', NORMAL), NORMAL etiketine atlar ve işlem normal şekilde devam eder. Eğer ikinci sayı 0 ise ve işlem operatörü bölme işareti ise, ekranda Sıfıra bölme hatası mesajını gösterir.

Bu kod parçası ayrıca EKRANA_YAZDIRMA etiketiyle başlar ve MOV R0, #0C0H komutuyla ekrandaki imleci ikinci satıra taşır. ACALL COMMAND komutu LCD üzerinde özel bir komutun çalıştırılmasını sağlar. Son olarak, MOV DPTR, #MSGERRO komutuyla Sıfıra bölme hatası mesajının bulunduğu bellek adresini DPTR kaydına yükler ve CLR C komutuyla taşma (carry) bayrağını temizler. Bu şekilde, ekrana hata mesajı yazdırma işlemi gerçekleştirilir.

	KOD	AÇIKLAMA
ASSEMBLY KOD		

```
SIRAYLA_KARAKTER_ALMA:          ; Sonraki karakteri almak için işlem başlangıcı
    MOV A, R7          ; R7 kaydındaki içeriği A kaydına kopyala
    MOVC A, @A+DPTR    ; A+DPTR adresindeki içeriği A'ya kopyala
    MOV R0, A          ; Sonucu R0 kaydına kopyala
    ACALL YAZDIR       ; LCD'ye yazdır
    INC R7             ; Eğer R7 sıfır değilse, R7'yi bir artır
    JNZ SIRAYLA_KARAKTER_ALMA          ; Eğer R7 sıfır değilse,
SIRAYLA_KARAKTER_ALMA etiketine atla
    RET                ; Alt programdan dön
```

Bu kod parçası, bir dizi karakteri sırayla alıp ekrana yazdırmak için kullanılır. R7 kaydındaki değeri A kaydına kopyalar ve DPTR ile A'nın toplamının adreslediği bellek hücresindeki değeri A'ya kopyalar. Sonucu R0 kaydına kopyalar ve LCD'ye yazdırmak için DISPLAY rutinini çağırır. R7'yi bir artırır ve eğer R7 sıfır değilse, SIRAYLA_KARAKTER_ALMA etiketine atlar, bu işlem dizideki sonraki karakteri almak için tekrarlanır. Eğer R7 sıfır ise, işlem sona erer ve alt programa döner.

	KOD	AÇIKLAMA

ASSEMBLY KOD	NORMAL:
	MOV R7, #100D ; R7'ye 100 değerini yükle
	CLR C ; Taşıma bayrağını temizle
	SUBB A, R7 ; A'dan R7'deki değeri çıkar-
	JC MENOR100; Eğer taşıma bayrağı taşınırsa, MENOR100 etiketine atla
	MOV A, R4; A'yı R4'e yükle (100'den büyük olduğu için 3 basamaklı sayı
	MOV B, R7 ; B'ye R7'yi yükle (100)
	DIV AB; A'yı B'ye böl, sonuç A'da, kalan B'de olur
	ADD A, #30H ; A'daki sayıyı ASCII karakterine dönüştür
	MOV R0, A; Karakteri ekrana göstermek için R0'a yükle
	ACALL YAZDIR; Karakteri ekrana göstermek için alt programı çağır
	MOV R4, B; B'deki kalanı (yüzler basamağı) R4'e yükle
	MOV A, B; B'deki kalanı (yüzler basamağı) R4'e yükle
	MOV R7, #10D ; R7'ye 10 değerini yükle
	MOV B, R7 ; B'ye R7'yi yükle (10)
	DIV AB ; A'yı B'ye böl, sonuç A'da, kalan B'de olur
	ADD A, #30H; A'daki sayıyı ASCII karakterine dönüştür
	MOV R0, A ; Karakteri ekrana göstermek için R0'a yükle
	ACALL YAZDIR; Karakteri ekrana göstermek için alt programı çağır
	MOV A, B; B'deki kalanı (onlar basamağı) A'ya yükl
	ADD A, #30H ; ASCII karakterine dönüştür
	MOV R0, A ; Karakteri ekrana göstermek için R0'a yükle
	ACALL YAZDIR; Karakteri ekrana göstermek için alt programı çağır
	CJNE R5, #00H, DECIMAL ; Eğer R5 0 değilse, DECIMAL etiketine atla
	RET; Alt programdan (subroutine) dön	

Normal etiketinin görevi, hesaplanan sayının üç basamaklı mı yoksa daha büyük olduğunu kontrol etmektir.

	KOD	AÇIKLAMA

ASSEMBLY KOD	KUCUK100:
	MOV R7, #10D; 10 sayısını R7 kaydediciye yükle
	CLR C; Taşıma (carry) bayrağını temizle
	MOV A, R4; R4'teki değeri A'ya yükle
	SUBB A, R7; R7'den A'daki değeri çıkar
	JC KUCUK10; Eğer taşıma bayrağı taşınmışsa (1 ise), MENOR10 etiketine atla
	MOV A, R4; A'daki değeri R4'ten tekrar A'ya yükle (R4 değişmedi)
	MOV B, R7; B'deki değeri R7'ye (10) yükle
	DIV AB; A'yı B'ye böl, sonuç A'da, kalan B'de olur
	ADD A, #30H; A'daki sayıyı ASCII karakterine dönüştür
	MOV R0, A; Karakteri ekrana göstermek için R0'a yükle
	ACALL DISPLAY; Ekrana karakteri göstermek için alt programı çağır
	MOV A, B; Kalanı (B) ekrana göstermek için A'ya yükle
	ADD A, #30H; Kalanı (B) ASCII karakterine dönüştür
	MOV R0, A; Karakteri ekrana göstermek için R0'a yükle
	ACALL DISPLAY; Ekrana karakteri göstermek için alt programı çağır
	CJNE R5, #00H, DECIMAL; Eğer R5 0 değilse, DECIMAL etiketine atla
	RET; Alt programdan (subroutine) dön	

Bu "KUCUK100" etiketi, hesaplanan sayının üç basamaklı olmadığı ve 100'den küçük olduğu durumu kontrol eder.

	KOD	AÇIKLAMA

ASSEMBLY KOD	KUCUK10:
	MOV A, R4; R4'teki değeri A'ya yükle
	ADD A, #30H; ASCII karakterine dönüştürmek için 30H (sayıların ASCII değeri) ekleyi
	MOV R0, A; Karakteri ekrana göstermek için R0'a yükle
	ACALL YAZDIR; Ekrana karakteri göstermek için alt programı çağır
	CJNE R5, #00H, DECIMAL; Eğer R5 0 değilse, DECIMAL etiketine atla
	RET; Alt programdan (subroutine) dön	

Bu "KUCUK10" etiketi, hesaplanan sayının üç basamaklı olduğu, ancak 100'den küçük ve 10'dan büyük olduğu durumu kontrol eder.

	KOD	AÇIKLAMA

ASSEMBLY KOD	TASMA:
	MOV R0, #0C0H; Ekranda mesajı ikinci satıra yazmak için R0'a 0C0H adresini yükler
	ACALL KOMUT; LCD ekranında belirli bir komutu yürütmek için alt programı çağırır
	MOV DPTR, #MSGERRO2; Hata mesajını göstermek için DPTR'yi MSGERRO2 adresine ayarlar
	CLR C; Taşıma bayrağını temizler
	MOV R7, #0D; Döngü için sayaç olarak R7'ye 0D değerini yükler	

Bu " TASMA" etiketi, hesaplama sırasında bir taşma (overflow) durumu oluştuğunda işlenir. Bu durum, sonuç bir byte'ın kapasitesini aşarsa veya bir hata durumunda ortaya çıkabilir.

	KOD	AÇIKLAMA

ASSEMBLY KOD	PROX2:
	MOV A, R7; R7'deki değeri A'ya yükle
	MOVC A, @A+DPTR; DPTR'nin gösterdiği bellek adresindeki veriyi A'ya yükle (MOVC ile)
	MOV R0, A; A'daki değeri R0'a yükle
	ACALL YAZDIR; Karakteri ekrana göstermek için alt programı çağır
	JZ FIM; Eğer A (R7'nin değeri) 0 ise, FIM etiketine atla
	INC R7; R7'yi bir arttır
	SJMP PROX2; PROX2 etiketine atla (döngüyü tekrar t)
	FIM:
	RET; Alt programdan (subroutine) dön	

Bu "PROX2" etiketi, bir döngüyü başlatmak ve belirli bir bellek adresindeki veriyi tekrar tekrar işlemek için kullanılır.

	KOD	AÇIKLAMA

ASSEMBLY KOD	ONLUK:
	MOV R0, #'.'; Nokta karakterini R0'a yükle
	ACALL YAZDIR; Karakteri ekrana göstermek için alt programı çağır
	MOV A, R5; R5'teki değeri A'ya yükle
	MOV B, #10D; 10 sayısını B'ye yükle
	MUL AB; A ve B'yi çarp ve sonucu A'ya yükle
	MOV B, R3; R3'teki değeri B'ye yükle
	DIV AB; A'yı B'ye böl, sonuç A'da, kalan B'de olur
	ADD A, #30H; A'daki sayıyı ASCII karakterine dönüştür
	MOV R0, A; Karakteri ekrana göstermek için R0'a yükle
	ACALL YAZDIR; Karakteri ekrana göstermek için alt programı çağır
	RET; Alt programdan (subroutine) dön	

" ONLUK " etiketi, bir ondalık sayının kalan kısmını ekrana göstermek için kullanılır.

	KOD	AÇIKLAMA

ASSEMBLY KOD	GECIKME:
	MOV 62, #2; 62 numaralı register'e 2 değerini yükle
	GECIKME 1:
	MOV 61, #250; 61 numaralı register'e 250 değerini yükle
	GECIKME 2:
	MOV 60, #250 ; 60 numaralı register'e 250 değerini yükle
	DJNZ 60, $; 60 numaralı register'deki değer sıfır olana kadar döngüyü devam ettir
	DJNZ 61, GECIKME 2; 61 numaralı register'deki değer sıfır olana kadar döngüyü devam ettir
	DJNZ 62, GECIKME 1; 62 numaralı register'deki değer sıfır olana kadar döngüyü devam ettir
	RET; Alt programdan (subroutine) dön
	DB - DEFINE BYTE
	HATA: DB 'HATA: DIV POR 0',0 Hata mesajı 2: Taşma hatası
	HATA2: DB 'TASMA HATASI!',0;Hata mesajı 2: Taşma hatası	

" GECIKME " etiketi, bir gecikme sağlamak için kullanılır. Bu gecikme, belirli bir süre boyunca programın belirli bir işlemi gerçekleştirmesini beklemek için kullanılır.