Ahmet Aycan Atak
Gökhan Seçinti

Mikrobilgisayar Laboratuvarı

2011-2012 Güz Dönemi

İtü-Eğit Kitleri İçin Deney Föyleri

Temmuz 2011

İstanbul Teknik Üniversitesi
Bilgisayar ve Bilişim Fakültesi

1

İTÜ-EĞİT Kitinin Tanıtılması

Bu deneyde, sonraki 3 deneyde kullanılması gereken İTÜ-EĞİT deney kitinin tanıtılması amaçlanmaktadır. İTÜ-EĞİT deney kiti temel olarak şu bileşenlerden oluşur;

- Motorola $MC6802$ merkezi işlem birimi
- 2 adet paralel iletişim arabirimi (PİA)
  - Gösterge ve tuş takımı için
  - Paralel yapılacak veri transferi için
- Gösterge ve tuş takımı
- Asenkron iletişim arabirimi (ASİA)
- Saltoku ve yaz/oku tipi bellekler
- PİA'ya bağlı 8 adet anahtar ve LED
- Adres, veri ve denetim yolu sürücüleri

Bu deney sonunda, deneyi yapan öğrencilerin İTÜ-EĞİT deney kitini kullanmayı, $MC6802$ işlemcisinin sahip olduğu adresleme yeteneklerini ve yine bu işlemci için simgesel dilde yazılmış kodları makine koduna çevirmeyi öğrenmiş olmaları amaçlanmaktadır.

1.1 Ön Bilgi

Bu bölümde İTÜ-EĞİT kitinin kullanımı ve kitin temelini oluşturan $MC6802$ işlemcisi için program yazımı ile ilgili gerekli bilgiler yer almaktadır.

1.1.1 İTÜ-EĞİT Kiti

İTÜ-EĞİT kiti, güç kablosu prize takıldığı zaman çalışır. Kitin çalışmasından sonra göstergeden okunacak H ibaresi kitin '$HAZIR$' konumuna geçtiğini ve kullanıma hazır olduğunu ifade etmektedir. Bundan sonra kit üzerinde bulunan tuş takımı ve göstergeler kullanılarak kit ile etkileşime geçilebilir.

Kit üzerinde gerçekleştirilebilecek temel işlemlerin kite program yükleme ve kite yüklenmiş programın adım adım ya da doğrudan çalıştırma olduğu söylenebilir. Bunların dışında İTÜ-EĞİT kiti dallanmalar sırasında sıçranacak adres miktarını hesaplama ve programın belli yerlerine duraksama noktaları (breakpoint) koyma gibi olanaklara da sahiptir.

Tuş takımının 2 parçadan oluştuğu kabul edilirse, $0 - F$ arası değerleri içeren kısım veri girişi için, 'M', 'S', 'Ex', 'Fc', 'R', 'Fs', 'G' ve 'T' tuşlarını içeren kısım ise farklı etkileşim fonksiyonları arasında seçim yapmak için kullanılmaktadır. Söz konusu işlemler ve bu işlemlerin gerçeklenmeleri aşağıdaki gibi verilebilir.

- **Program Yazma:** Makine koduna çevirilmiş programın deney kitine yüklenmesi için, kit '$HAZIR$' konumundayken, programın yazılacağı adres tuş takımından girilir [1]. Programın yazılacağı adresin tuş takımından girilmesi sonrası, 'M' tuşuna basılarak o adresin bulunduğu bellek gözüne gidilir. Bundan sonra bellek gözünün içeriği tuş takımından veri girişi yapılarak değiştirilebilir. Bir sonraki bellek gözünün içeriği değiştirilmek istendiğinde 'G' tuşu ile, bir önceki bellek gözünün içeriği değiştirilmek istendiğinde ise 'M' tuşu ile farklı bellek adresleri arasında geçiş yapılabilir.
- **Program Çalıştırma:** Makine koduna çevirilmiş program deney kitine yüklendikten sonra, kit '$HAZIR$' konumundayken[2], programın başlangıç adresi tuş takımından girilir ve 'G' tuşuna basılır. Bundan sonra program çalışacak ve program sayacı programın bitmesi gereken yere geldiğinde duracaktır.
- **Adım Sayısı Hesaplama:** Dallanma komutlarından sonraki bellek gözlerine yazılması gereken sıçrama miktarlarının hesaplanması İTÜ-EĞİT kiti üzerinde yapılabilir. Bunun için program yazma modunda, sıçrama miktarının yazılacağı bellek gözüne gelinip 'Fs' tuşuna basılır. Göstergede 'A' ibaresi okunduktan sonra sıçranması gereken adres tuş takımından girilir. Adres girildikten sonra 'G' tuşuna basıldığında kit sıçrama miktarını göstergede yazar. 'G' tuşuna bir kez daha basıldığı takdirde bu sıçrama miktarı yazılması gereken bellek gözüne yazılır.
- **Duraksama Noktası Koyma:** Programı önceden belirlenmiş bir noktaya kadar çalıştırmak için İTÜ-EĞİT kiti içerisinde duraksama noktaları tanımlanabilir. Bunun için kit '$HAZIR$' konumundayken 'Fs' tuşuna bir kez basılır. Hemen ardından 'T' tuşuna basılır. Göstergenin en sağında görünen sayı kit içerisinde tanımlanmış duraksama noktalarının sayısıdır. Bundan sonra duraksama noktasının adresi girilir ve 'Fs' tuşuna basılır. Bu noktada duraksama noktası tanımlanmış olur ve kit içerisinde tanımlanmış duraksama noktalarının sayısını belirten ifade 1 artar. İstenirse aynı şekilde daha çok duraksama noktası tanımlanabilir. Çalıştırılan programlar bu

[1] OKU/YAZ bellekler $\$0000 - \$007F$ ve $\$4000 - \$5FFF$ adresleri arasında konumlanmıştır. Programlarınızı bu adresler arasına yazmaya dikkat ediniz.

[2] Kiti kullanırken herhangi bir anda 'Ex' ya da '$Reset$' tuşlarına basmak kiti tekrar '$HAZIR$' konumuna getirecektir.

duraksama noktalarına denk geldiklerinde çalışmayı o anki durumlarıyla durduracaktır[3].

- **Adım Adım Çalıştırma** Deney kiti hazır konumundayken 'R' tuşuna basılır. 'PS' (program sayacı) kütüğüne programın başlangıç adresi yazılır ve her adım için 'T' tuşuna basılarak program adım adım çalıştırılır. Her adımda akümülatörlerin ve kütüklerin değerleri 'M' ve 'G' tuşları kullanılarak görüntülenebilir. Hatalı programların hatalı oldukları yerler, adım adım çalıştırma işlemi yapılarak kolaylıkla bulunabilir.

1.1.2 MC6802 İşlemcisi

$MC6802$ işlemcisi 2 adet akümülatöre (A ve B - 8'er bit), 1 adet program sayacına (PS - 16 bit), 1 adet sıralama kütüğüne (SK[4] - 16 bit), 1 adet yığın göstergesine (YG - 16 bit) ve 1 adet durum kütüğüne (DU - 8 bit) sahiptir. $MC6802$'nin komut setinde, komutların ne anlama geldikleri ve etkileri bu kütükler kullanılarak ifade edilmiştir.

MC6802 işlemcisi 5 farklı adresleme yeteneğine sahiptir. Her bir adresleme tipine örnek Tablo 1.1'de verilmiştir. Tablo 1.1'deki örneklerle paralel olarak, bu adresleme yöntemlerinin işlevleri kısaca şu şekilde açıklanabilir.

- **İvedi Adresleme (I):** Kütüklerin doğrudan veri ile ilişkilendirildiği adresleme çeşididir. 'A akümülatörüne $55 yüklemek' ya da 'Sıralama kütüğünün içeriğini $4500 ile karşılaştırmak' gibi işlemler bu adresleme tipine örnek olarak verilebilir. Makine kodu olarak, komutun karşılığının yazıldığı bellek adresini takip eden adreslere, ivedi adreslemede kullanılacak değerler yazılır.
- **Ön Sayfalı Adresleme (D):** $00-$7F arasındaki bellek adreslerinin içerikleri ile yapılacak işlemler için tanımlanmıştır. '$10 adresindeki değeri A akümülatörüne yüklemek' ya da 'B akümülatörünü $30 adresindeki değer ile toplamak' gibi işlemler bu adresleme tipine örnek olarak verilebilir. Komutun makine koduna çevrilmesinden ardından parametre olarak 8 bitlik bellek adresi kullanılır.
- **Sıralı Adresleme (S):** 16 bitlik sıralama kütüğünde tutulan adres ile ilgili işlemler yapılacağı zaman kullanılır. 'A akümülatörünün içeriğini sıralama kütüğündeki adresin 2 adres sonrasına yazma' ya da 'B akümülatörünü sıralama kütüğündeki adresin içerdiği değerle karşılaştırma' gibi işlemler bu adresleme tipine örnek olarak verilebilir. Komutun makine koduna çevrilmesinin ardından kayıklık miktarı parametre olarak bir sonraki bellek adresinde kullanılır.
- **Dolaylı Adresleme (G):** 16 bitlik bellek adresleri ile yapılacak işlemlerde kullanılır. Bu işlemlere örnek olarak 'A akümülatörüne $4100 bellek adresindeki değeri atama' ya da 'SK kütüğünün içeriğini $4100 bellek adresin-

[3] Göstergede beliren program sayacı dışındaki kütüklerin ve akümülatörlerin değerleri 'M' ve 'G' tuşarı kullanılarak görülebilir

[4] İTÜ-EĞİT kitinde bu kütük SI şeklinde isimlendirilmiştir.

deki değer ile karşılaştırma'[5] verilebilir. Makine koduna çavirme işlemi yapılırken, komutun makine kodu karşılığını, 16 bitten oluşan bellek adresi parametre olarak izlemelidir.

- **Doğrudan Adresleme (L):** Parametre almayan komutlar bu adresleme tipine dahildir. Komutun etkilediği kütükler belli olduğu için parametre kullanılmaz. 'B akümülatörünün içeriğini temizleme' ya da 'altprogramdan dönme' gibi işlemler bu adresleme tipine dahil olan kullanımlara örnek olarak verilebilir.

Tablo 1.1. Farklı adresleme tipleri için örnekler

Simgesel Dilde Komut	Makine Kodu Karşılığı	Adr. Tipi
LDAA $55	86 55	I
CPX $4500	8C 45 00	I
LDAA [$10]	96 10	D
ADDB [$30]	DB 30	D
STAA [SK+02]	A7 02	S
CMPB [SK+00]	E1 00	S
LDAA [$4100]	B6 41 00	G
CPX [$4100]	BC 41 00	G
CLRA	4F	L
RTS	39	L

Bu bölümde İTÜ-EĞİT deney kitinin kullanımına ve $MC6802$ işlemcisinin programlanmasına dair verilen bilgiler, sonraki 3 deneyin gerçeklenmesi için gerekecek bütün bilgi gereksinimini kapsamaktadır. Bunun dışında, her deney başında, deneye özel bilgiler paylaşılacaktır.

1.2 Deney

Bu deneyde sırasıyla, simgesel dilde verilmiş örnek programın makine koduna çevrilmesi ve makine kodunda verilmiş programın simgesel dile çevrilmesi istenmektedir.

1.2.1 Simgesel Dilden Makine Koduna

Aşağıda, $MC6802$ simgesel dilinde verilmiş kodu makine koduna çevirip, İTÜ-EĞİT deney kitine yükleyiniz. Kodu makine koduna çevirirken karşılaştığınız dallanma komutlarında, sıçranacak adım miktarını bir önceki bölümde anlatılan yöntemle, İTÜ-EĞİT kiti kullanarak hesaplayınız.

[5] 16 bitlik SK kütüğü, $4100 yüksek anlamlı, $4101 düşük anlamlı kısım olarak 16 bitlik ifade ile karşılaştırılacaktır.

```
1         LDX     $0000
2         CLRA
3         CLR     [$0010]
4         CLR     [$0011]
5         LDAA    [$4500]
6         STAA    $0010
7  D1:    LDAB    [$0010]
8         CMPB    $00
9         BEQ     C1
10        STAA    $0011
11 D2:    LDAB    $0011
12        CMPB    $00
13        BEQ     C2
14        INX
15        DEC     $0011
16        BRA     D2
17 C2:    INC     [$0010]
18        BRA     D1
19 C1:    STX     $4500
20        SWI
```

Programı başarılı bir şekilde çalıştırdıktan sonra, kodu yorumlayınız ve çıktılarını kontrol ediniz. Programı değiştirmeden farklı çıktılar alabilmek için gerekli düzenlemeleri yapınız. Sonucun beklediğiniz gibi olup olmadığını tartışınız.

1.2.2 Makine kodundan Simgesel Dile

Aşağıda bir programın, $4000 adresinden başlayarak yazılmış makine kodu karşılığı verilmiştir.

```
1  ($4000)    4F
2  ($4001)    CE 45 00
3  ($4004)    8C 45 0A
4  ($4007)    27 0F
5  ($4009)    A6 00
6  ($400B)    81 00
7  ($400D)    2D 03
8  ($400F)    08
9  ($4010)    20 F2
10 ($4012)    43
11 ($4013)    4C
12 ($4014)    A7 00
13 ($4016)    20 EC
14 ($4018)    3F
```

Bu programı simgesel koddaki karşılığına çevirerek ne yaptığını anlamaya çalışınız. Daha sonra programı çalıştırarak beklediğiniz sonuçları alıp almadığınızı tartışınız.

Makine kodu verilmiş programda $4016 adresindeki dallanmanın Tablo 1.2'de verilen adreslere yapıldığını düşünerek, İTÜ-EĞİT kiti yardımıyla sıçrama miktarlarını hesaplayınız. Bulduğunuz sonuçlar yardımıyla tablo 1.2'yi doldurunuz.

Tablo 1.2. $4016'dan dallandığımızda yapılması gereken sıçrama miktarları

Dallanılacak Adres	Sıçrama Miktarı
$402F	
$410F	
$46FA	

Deneyin geneli boyunca anlamadığınız ve kafanıza takılan yerleri deney gözetmenlerine sormaktan çekinmeyiniz.

1.3 Rapor

Bu deneyin raporunu şu başlıklardan oluşan herhangi bir formatta, düzenli ve titiz bir şekilde hazırlayınız.

- **Deneyin İçeriği:** Bu bölümde deney boyunca neler yapıldığı kısaca tanıtılacaktır.
- **Deneyde Verilen Programlar:** Bu bölümde, üzerinde çalışılması için verilmiş örnek programların ne işe yaradığından bahsedilecek ve bu kodların C dilinde yazılmış karşılıkları verilecektir [6]. Her iki program için de C kodlarıyla beraber akış diyagramı verilmelidir.
- **Sonuç ve Yorumlar:** Bu bölümde deneyin, en başta belirlenen amaçlara ulaşıp ulaşmadığından bahsedilecek ve varsa deneyin işlenişi ve içeriğiyle ilgili yorumlar yapılacaktır.

Özensiz, baştan savma hazırlanmış raporlar dikkate alınmayacak olup kopya raporlar[7] negatif puan ile cezalandırılacaktır.

[6] Değişken isimlerinin kütük isimleriyle paralel olmasına dikkat ediniz.
[7] Bir şeklin birkaç raporda ortak kullanımından, bir ifadenin değişik raporlarda aynı şekilde bulunmasına kadar her şey kopyadır.

2

MC6802 İşlemcisinde Program Yazma

Bu deneyde, bir önceki deneyde tanıtımı yapılmış olan $MC6802$ işlemcisi kullanılarak program geliştirme ve gerçekleme konusunda pratik yapılması planlanmaktadır. Bu amaçla ön bilgi ve deney kısımlarında verilen bilgiler ışığında aşağıdaki algoritmaların $MC6802$ işlemcisi için gerçeklenmesi istenmektedir.

- Bit bazında şifreleme işlemi
- Bellek sınama
- Kabarcık (ya da Baloncuk) Sıralama (Bubblesort)

Deney sonunda, deneyi yapan öğrencilerin $MC6802$ işlemcisinin simgesel dilini kullanarak program geliştirme ve yazma konusunda yeterli teknik donanıma sahip olması amaçlanmaktadır.

2.1 Ön Bilgi

Bu bölümde Vigenere şifreleme ve kabarcık sıralama algoritmaları hakkında gerekli olan bilgiler yer almaktadır.

2.1.1 Bit Bazında Şifreleme

Bit bazında şifreleme işlemi, eldeki verinin bitlerinin manipüle edilmesi ile gerçekleştirilir. Bit bazında şifreleme için yaygın kullanılan yöntemlerden birisi, verinin bir anahtar veri ile XOR işlemine sokulmasıdır. Şifrelenmiş veri aynı anahtarla tekrar XOR işlemine sokulduğunda şifrelenmiş veri çözülür.

Deneyde gerçeklenmesi istenilen şifreleme yöntemi şu şekilde çalışmaktadır,

- Verinin ilk dört biti ile son dört biti yer değiştirir (ABCDEFGH -¿ EFGHABCD)
- Elde edilen verinin bitleri ikişerli olarak yer değiştirir (EFGHABCD -¿ FEHGBADC)

- Son olarak elde edilen veri, başka bir 8 bitlik anahtar ile XOR işlemine sokulur ($EFGHABCD \oplus KLMNOPQR$)

Yukarıda verilen işlemlerin, şifreli koda, aynı anahtar ile tersten uygulanmasıyla şifreli veri çözülebilir.

2.1.2 Bellek Sınama

Adresi m olarak verilmiş bir bellek alanının yazılabilir olup olmadığı şu şekilde test edilebilir.

- m adresine $\$AA$ yazılır.
- m adresinden $\$AA$ okunuyorsa bir sonraki adıma geçilir. Aksi halde m adresli bellek gözüne yazma izni yoktur.
- m adresine \$55 yazılır.
- m adresinden \$55 okunuyorsa, m adresli bellek gözüne yazma izni vardır. Aksi halde bu bellek gözüne yazma izni yoktur.

Yukarıda verilen yöntem ile bellek gözünün her bir bitine 1 ve 0 değerleri yazılıp okunmuş olur. Böylece bellek adresi içerisinde meydana gelmiş problemler de ortaya çıkarılabilir.

2.1.3 Kabarcık Sıralama

n elemanlı sıralanabilir bir A dizisi[1], kabarcık sıralama yöntemiyle şu şekilde sıralanabilir.

```
1 for i=1 to n
2   for j=1 to n-1
3     if A[j]>A[j+1]
4       A[j] ile A[j+1]'in yerini degistir.
5     endif
6   endfor
7 endfor
```

Yukarıda verilen örnek kod parçasında, dizinin ilk elemanının indisinin 1, son elemanının indisinin ise n olduğu varsayılmıştır.

2.2 Deney

Deney sırasında, ön bilgiler kısmında bahsedilmiş programlar, bu kısımda belirtilen özellikler de göz önüne alınarak gerçeklenecektir.

[1] Sıralanabilir bir dizi, karşılaştırılabilir elemanlara sahip olmalıdır.

2.2.1 Şifreleme

Ön bilgiler kısmında bahsedilen şifreleme yöntemini şu nitelikleri göz önüne alarak gerçekleyiniz.

- $10 adresinden şifrelenecek veri okunacaktır.
- $11 adresinde şifrelemede kullanılacak anahtar kelime yer alacaktır.
- Sonuç $12 adresine yazılacaktır.

Programı kite geçirdikten sonra yazdığınız kodun çalışıp çalışmadığını, elle hesapladığınız bazı veri-anahtar ikilileri için test ediniz.

2.2.2 Bellek Sınama

Bellek sınama için ön bilgiler kısmında verilen yöntemi, verilen bir bellek aralığı için, aşağıda verilen özellikleri de göz önüne alarak kodlayınız.

- Sınanacak bellek aralığının başlangıç adresi $10(−$11) adresinde yer alacaktır.
- Sınanacak bellek aralığının bitiş adresi $12(−$13) adresinde yer alacaktır

Program sonlandığında bellek adresinin yazılabilir olup olmadığı, A akümülatörü üzerinde taşınacak değer ile belirtilecektir[2]. Programı kite yükledikten sonra, bir önceki deneyde sözü edilen farklı adres aralıkları için programınızı test ediniz.

2.2.3 Kabarcık Sıralama

Ön bilgiler kısmında sözü edilen kabarcık sıralama yöntemini, aşağıda verilen özellikleri de sağlayacak şekilde gerçekleyip, kite giriniz.

- Sıralanacak dizinin başlangıç adresi $10(−$11) adresinde yer alacaktır.
- Sıralanacak dizinin eleman sayısı $12 adresinde yer alacaktır.

Programı farklı dizi adresleri ve eleman sayıları için test ediniz. Program çalışmayı sonlandırdığında, üzerinde çalışılan dizinin sıralanmış olması gerekmektedir.

2.3 Rapor

Bu deneyin raporunu şu başlıklardan oluşan herhangi bir formatta, düzenli ve titiz bir şekilde hazırlayınız.

[2] Program sonlandığında A akümülatöründe 11 varsa bellek adres aralığına yazılabilir, 00 varsa yazılamaz

- **Deneyin İçeriği:** Bu bölümde deney boyunca neler yapıldığı kısaca tanıtılacaktır.
- **Deney:** Deney sırasında hazırladığınız programları, akış diyagramları ile birlikte bu bölümde veriniz.
- **Sonuç ve Yorumlar:** Bu bölümde deneyin, en başta belirlenen amaçlara ulaşıp ulaşmadığından bahsedilecek ve varsa deneyin işlenişi ve içeriğiyle ilgili yorumlar yapılacaktır.

Yukarıdaki başlıklara ek olarak raporda, konu bütünlüğünü koruyacak şekilde istediğiniz herhangi bir meseleye ek başlık açabilirsiniz[3].

[3] Deneyde karşılaşılan problemler, bu problemlerin nasıl çözüldüğü, vs. gibi başlıklar için deney anının verimli şekilde not almanız önerilir.

3

Altprogram ve Yığın İşlemleri

Bu deneyde, İtü-Eğit deney kiti kullanılarak altprogram ve yığın kullanımı konusunda uygulama yapılması planlanmaktadır. Deneyden önce,

- Mikroişlemcilerde altprogram çağrısı yapılırken sistem yığınının üstlendiği işlevlerin
- Altprograma parametre aktarımın nasıl yapıldığının
- Altprogramların nasıl çağrıldığının

hatırlanmasında yarar vardır. Bu deney 3 bölümden oluşmaktadır ve her deney için gerekli ön bilgiler deneyle aynı başlık altında verilmiştir. Deney sonunda, deneyi yapan öğrencilerin yığının işleyişi, altprogram-yığın etkileşimi gibi konuları kavramış olması amaçlanmaktadır.

3.1 Deney

Bu bölümde, deney esnasında gerçeklenmesi istenilen programlarla ilgili bilgiler, istenen programların ne oldukları bilgisiyle beraber verilecektir.

3.1.1 Altprogram - Yığın Etkileşimi

Deneyin ilk kısmında aşağıda 6802 simgesel dilinde hazır olarak verilmiş olan program kodu yazılarak adım adım çalıştırılacak ve her adımda SK'nin içeriği incelenecektir. Aşağıda simgesel dilde verilmiş kodu, makine koduna çevirerek İtü-Eğit kitine giriniz.

```
1         LDS     $5F00
2         TSX
3         JSR     ALT1
4         SWI
5 ALT1:   TSX
6         JSR     ALT2
```

```
7           RTS
8 ALT2:     TSX
9           SWI
```

Yukarıda verilen programı İtü-Eğit kitine girdikten ve doğru çalıştığından emin olduktan[1] sonra adım adım çalıştırarak tablo 3.1'i doldurunuz. Tablonun ölçümler için eksik kalması veya fazla olması gibi durumları göz ardı ederek ölçümleri yapınız.

Tablo 3.1. İlk deney için doldurulacak tablo

Adım	Sıralama K.	Program S.	Adım	Sıralama K.	Program S.

Tablo 3.1'i raporunuza ekleyiniz ve tablodaki değerlerin değişimleri ile ilgili yorumlarınızı rapora eklemek için not alınız. Raporda yer alacak yorumsuz sonuçların değerlendirilmeyeceğini göz önüne alınız.

3.1.2 4 Bitlik Sayıların Çarpımı

Deneyin ikinci kısmında 4 bitlik sayılar için çarpma işlemi yapan bir altprogramın gerçeklenmesi istenmektedir. Şu durumda, sizden sadece altprogramın içeriğini verilen çarpma algoritmasına göre yazmanız beklenmektedir. Altprogram yığın üzerinden iki adet 8 bitlik parametre alacak ama yalnızca en düşük anlamlı 4 bitlik kısımlarını kullanacaktır. Altprogram dönüş değerini A akümülatörü üzerinden göndermelidir.

```
1           LDS     #003FH
2           LDAA    0000H
3           PSHA
4           LDAA    0001H
5           PSHA
6           JSR     CARP
7           INS
8           INS
9           SWI
10
11 CARP:    ;
12          ; burada alt programi gercekleyiniz...
13          ;
```

[1] Program doğrudan çalıştırıldığında sonsuz döngüye giriyor mu? Bu sorunun cevabı evet ise dallanma hesaplarını kontrol ediniz.

'CARP' altprogramında gerçeklemeniz gereken algoritmanın sözde kodu aşağıdaki gibidir. Sözde koddan da görüldüğü gibi *par*1 ve *par*2 çarpılacak sayılardır. Çarpım işleminin sonucu *sonuc* değişkenindedir.

```
1         par1 ve par2 carpilacak sayilar
2         sonuc := 0
3         for basamak sayisi kadar
4                 par1'i saga kaydir
5                 if elde
6                         sonuc := sonuc + par2
7                 endif
8                 par2'yi saga kaydir
9         endfor
```

Yazdığınız altprogram içerisinde kullanacağınız değişkenleri saklamak için statik adresler kullanabilirsiniz.Yazdığınız kodun anlaşılabilmesi açısından kullandığınız bu adreslere ilişkin açıklamaları deney raporuna eklemeniz gerekmektedir.

3.1.3 Rekürsif Çalışan Altprogram

Deneyin üçüncü kısmında rekürsif çalışan bir altprogram yazmanız istenmektedir. Altprograma parametre yığın üzerinden gönderilecek ve altprogram dönüş değer A akümülatörü üzerinden yapılacaktır. Fonksiyonun matematiksel ifadesi aşağıda verilmiştir.

$$f(x) = \begin{cases} 1 & \text{eger x} = 0 \\ x + f(x-1) & \text{aksi takdirde} \end{cases} \quad (3.1)$$

Burada dikkat edilmesi gereken kısım altprogramın çağrıldığı noktada kullanılan ve anlamlı değerler içeren kütüklerin içeriklerinin, içerikler saklanmadan altprogram içerisinde kütüklerin tekrar kullanılmasıyla kaybedilmesidir.

Altprogramda ilk olarak, altprogram içerisinde kullanılacak kütüklerin içerikleri yığına atılmalı ve altprogram sonlanırken yığından geri çekilerek, altprogramın çağrıldığı noktada kütük değerlerinin bozulması engellenmelidir. Yukarıda anlatılan işlemleri gerçekleştiren altprogram şablonu aşağıda verilmiştir. Deneyin bu kısmında sizden beklenen kodda işaretlenmiş olan alana yukardaki verilmiş olan matematiksel ifadeyi gerçekleştiren kodu yazıp ekleyerek fonksiyonu çalıştırmanızdır. Ekleyeceğiniz kodun sonunda dönüş değerinin A akümülatörüne yazılmış olması gerekmektedir.

```
1 ;Ana program
2         LDS     #0050H          ;YG icin ilk degeri ata
3         LDAA    #04
4         PSHA                    ;parametre gonder
5         JSR     FACT
6         INS                     ;parametreyi temizle
```

```
7           SWI                         ;program sonu
8  ;Alt program
9  FACT:    PSHB            ;AccB icerigini sakla
10          STX     0000H   ;SK icerigini sakla
11          LDAB    0000H
12          PSHB
13          LDAB    0001H
14          PSHB
15          TSX             ;SK <- YG
16
17          LDAB    5,X     ;AccB  <-  <SK+05>
18                          ;AccB ye SK yi kullanarak yigin ile
19                          ;gonderilen parametrenin degerini al
20
21 ;Verilen Fonksiyonu Gerceklestirecek Olan Kod Buraya Eklenecek
22
23 F_SON:   PULB            ;Altprogram sonu
24          STAB    0001H   ;Yigindan parametrelerin geri alinmasi
25          PULB
26          STAB    0000H
27          LDX     0000H
28          PULB
29          RTS
```

İstenilen kod kısmını ekleyip altprogramı çalıştırdıktan sonra altprogramı 2 değeri için adım adım çalıştırın ve PC, YG, AccA'nin değerlerini tablo 3.2'yi doldurun. Yığın göstergesinin en düşük değeri için yığının görüntüsünü çıkarın ve doldurmuş olduğunuz tablo ile beraber raporunuza ekleyin.

Tablo 3.2. Üçüncü deney için doldurulacak tablo

Adım	Yığın G.	Program S.	AccA	Adım	Yığın K.	Program S.	AccA

Elde ettiğiniz bu veriler ile ilgili mutlaka raporunuza yorum yazın, yazdığınız raporda yalnızca sayısal değerlerin ve kodların olması bir anlam ifade etmeyecektir.

3.2 Rapor

Bu deneyin raporunu şu başlıklardan oluşan herhangi bir formatta, düzenli ve titiz bir şekilde hazırlayınız.

- **Deneyin İçeriği:** Bu bölümde deney boyunca neler yapıldığından kısaca bahsedilecektir.
- **Deney:** Deney sırasında hazırladığınız programları, açıklama satırları ile birlikte bu bölümde veriniz.
- **Sonuçlar:** Bu bölümde deney esnasında elde ettiğiniz sonuçları yine deneyde istenen çıkarımları ve yorumları yaparak veriniz.
- **Yorumlar:** Deneyin başlangıçta belirlenen amaçlara ulaşıp ulaşmadığı, yararlı olup olamdığı bu bölümde tartışılacaktır.

Yukarıdaki başlıklara ek olarak raporda, konu bütünlüğünü koruyacak şekilde istediğiniz herhangi bir meseleye ek başlık açabilirsiniz.

4

Paralel İletişim Arabirimi

PİA, bilgisayarın çevre birimleri ile paralel iletişimini sağlayan birimdir. Bu deneyde, PİA'nın kullanımını öğretmek üzere bazı uygulamalar yapılacaktır. Deneyde amaçlananlar şu şekildedir.

- Paralel iletişim arabiriminin kullanımın öğrenilmesi
- Yedi kollu göstergenin kullanımın öğrenilmesi

Bu deney sırasında PİA'dan alınan çıkışlar CADET ve İtü-Eğit üzerinde bulunan LED'ler kullanılarak kontrol edilecektir.

4.1 Ön Bilgi

İTÜ-Eğit'te PİA deneylerin gerçekleyebilmek üzere bir PİA bulunmaktadır. PİA olarak kullanılan kırmığın adı $MC6821$'dir. $MC6821$ PİA kırmığı içinde, özdeş iki iskele (port) bulunmaktadır ve bunlar PİA-A (A iskelesi) ve PİA-B (B iskelesi) olarak adlandırılmaktadır. Her PİA'da iskeleler, yönlendirici ve durum/denetim kütüğü bulunmaktadır. Aynı kırmık içinde iki iskele olması nedeniyle kütüklerin sayısı altıdır. MC6821'de temel adres seçiciye ek olarak, Şekil-1'de görüldüğü gibi iki adet kütük seçici girişi (RS0, RS1) bulunmaktadır. Bu iki seçici ile altı kütüğün nasıl seçildiği tablo 4.1'de açıklanmıştır.

Tablo 4.1. Pia içerisindeki kütükler ve bu kütüklerin seçilmesi

RS1	RS2	CA2	CB2	Seçim
0	0	1	X	İskele-A
0	0	0	X	Yönlendirici-A
0	1	X	X	Durum/Denetim Kütüğü-A
1	0	X	1	İskele-B
1	0	X	0	Yönlendirici-B
1	1	X	X	Durum/Denetim Kütüğü-B

CA2, PİA-A'nın Durum/Denetim Kütüğünün üçüncü bitidir. Benzer şekilde, CB2 PİA-B'nin Durum/Denetim Kütüğünün üçüncü bitidir. Yukarıdaki tablodan anlaşılacağı gibi, PİA'nın iskele ve yönlendiricisinin adresleri aynıdır. İskele ya da yönlendiricinin seçimi, Durum/Denetim kütüğünün üçüncü bitine 0 ya da 1 yazılarak belirlenmektedir. Bu bilgilerden yola çıkarak PİA içerisindeki kütüklerin adresleri tablo 4.2'deki gibi verilebilir.

Tablo 4.2. Pia içerisindeki kütüklerin adresleri

Kütük	Adres
İskele-A	$8300
Yönlendirici-A	$8300
Durum/Denetim Kütüğü-A	$8301
İskele-B	$8302
Yönlendirici-B	$8302
Durum/Denetim Kütüğü-B	$8303

MC6821'in iskelesinde bulunan her kapı, alıcı ya da verici olarak kullanılabilmektedir. Her bir kapının alıcı mı yoksa verici mi olduğu yönlendirici ile belirlenmektedir. Bir kapıyı verici yapmak için yönlendiricinin ilgili gözesine 1, alıcı yapmak için 0 yazmak gerekir.

4.2 Deney

Bu deneyde sırasıyla temel giriş çıkış işlemleri ile yedi kollu gösterge uygulamaları gerçeklenecektir.

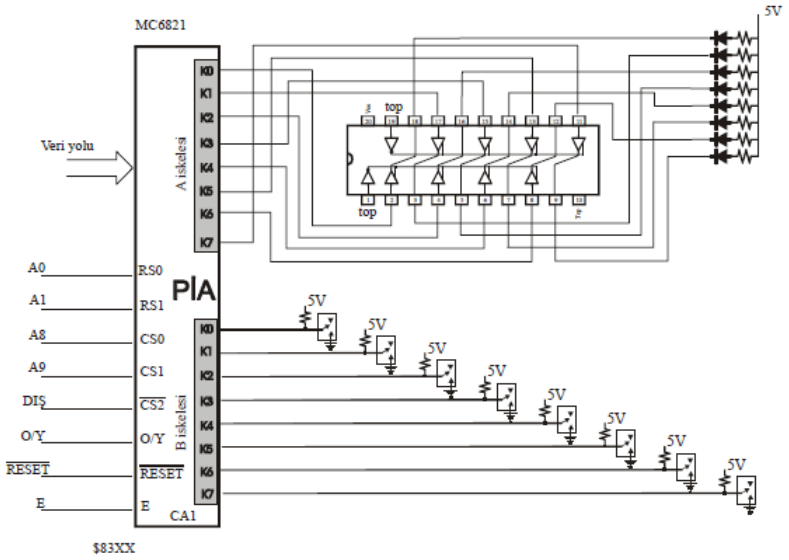
4.2.1 Temel Giriş Çıkış İşlemleri

İTÜ-Eğit'te kullanıcı PİA'sının iskele-B'sine bağlı 8 adet anahtar ve iskele-A'sına bağlı 8 adet LED (iskele ile LED ler arasında bir sürücü devre yer almaktadır) bulunmaktadır. PİA'nın LED'ler ve anahtarlarla yaptığı bağlantılar şekil 4.1'de gösterilmiştir.

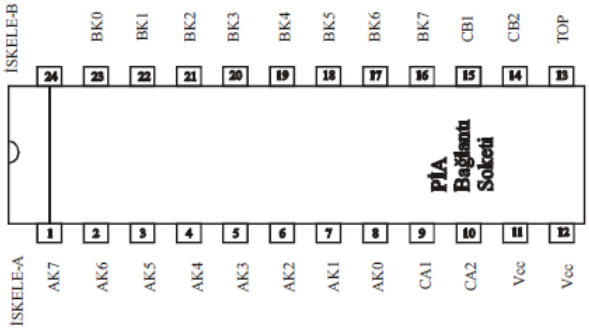
Anahtarların konumlarını okuyan ve anahtarların konumlarına bağlı olarak LED'leri yakan bir program yazınız. Bu deneyi yapabilmek için, ilk aşamada PİA-B'nin tüm kapılarını alıcı ve PİA-A'nın tüm kapılarını verici olacak şekilde koşullayınız.

4.2.2 Yedi Kollu Gösterge Deneyi

İTÜ-Eğit'te bulunan kullanıcı PİA'sı, değişik deneyleri yapabilmek amacıyla tasarıma eklenmiştir. Bu PİA'nın iskele ayakları yanında bulunan konnektöre



Şekil 4.1. Pia ile anahtarların ve LED'lerin bağlantısı



Şekil 4.2. Pia konnektörünün ayakları

aktarılmıştır. PİA'nın kapılarının konnektörün hangi ayaklarına bağlı olduğu şekil 4.2'de gösterilmiştir.

Yedi kollu göstergeyi bağlamak için 7449 tümdevresi sürücü olarak kullanılacaktır. Bu tümdevrenin kataloğu ekte verilmiştir. Burada dikkat edilmesi gereken bu devreyi kullanırken Vcc ve GND bağlantılarının da İTÜ-Eğit üzerindeki konnektörden alınmasıdır. Deneyde 7449'un bağlantıları kullanıcı PİA'sı üzerindeki A iskelesinden yapılacaktır. B iskelesine bağlı anahtarlar giriş olarak kullanılacaktır. Burada istenilen B iskelesinden en düşük anlamlı 4 bitinin belirtiği sayının yedi kollu gösterge üzerinde okunmasıdır.

4.3 Rapor

Bu deneyin raporunu aşağıda verilen başlıklardan oluşan herhangi bir formatta, düzenli ve titiz bir şekilde hazırlayınız.

- **Deneyin İçeriği:** Bu bölümde deney boyunca neler yapıldığından kısaca bahsedilecektir.
- **Deney:** Deney sırasında hazırladığınız programları ve kurduğunuz devrelerin çizimlerini bu bölümde veriniz..
- **Sonuçlar:** Bu bölümde deney esnasında elde ettiğiniz sonuçları, deneyi başarılı bir şekilde gerçekleyip gerçekleyemediğinizi, eğer deneyde başarılı olamadıysanız muhtemel nedenleri belirtiniz.
- **Yorumlar:** Deneyin başlangıçta belirlenen amaçlara ulaşıp ulaşmadığı, yararlı olup olamdığı bu bölümde tartışılacaktır.

Yukarıdaki başlıklara ek olarak raporda, konu bütünlüğünü koruyacak şekilde istediğiniz herhangi bir meseleye ek başlık açabilirsiniz.