SAKARYA ÜNİVERSİTESİ
Bilgisayar ve Bilişim Bilimleri Fakültesi
Bilgisayar Mühendisliği Bölümü

Mikroişlemcili Sistemler ve Laboratuvarı

8.HAFTA

Amaçlar

- Assembly dilinin genel özelliklerini tanımak
- 8051 komut setinin kullanımı hakkında bilgi sahibi olmak
- Assembly dilinde örnek uygulama yazabilmek

Assembly Dili ve Özellikleri

- Her bir komut, gerçekleştirdiği işleve karşılık gelen İngilizce kelimenin kısaltması (mnemonic-anımsatıcı) ile ifade edilir.
- Her bir kısaltma, makine dilindeki farklı bit dizisine karşılık gelmektedir.
- MOV A, #55 ;Akümülatöre 55 değerini yükle
- Assembly dilini kullanarak uygulama geliştirmek makine diline (ikili sayı sistemine) göre daha kolaydır.
- C, Basic, Pascal vb. gibi yüksek seviyeli dillerde program yazmaya nazaran daha zordur.

Assembly Dili ve Özellikleri

Assembly Dilinin Avantajları

- Donanım hakkında daha fazla detay bilmeyi gerektirir. Bu bir programcı için dezavantaj gibi görülse de üzerinde çalışılan donanım hakkında tasarımcıya önemli bilgi birikimi sağlar.

- Özel donanım ihtiyaçları üzerinde daha fazla kontrol sağlar.

- Yüksek seviyeli dillere göre daha küçük, daha az yer kaplayan ve daha hızlı icra edilebilir kodlar üretilebilir.

Program Formatı

Etiket	İşlem Kodu (Komut)	İşlenen (Operand)	Açıklama
Basla:	MOV	A, #77h	;Aküye 77_{16} değerini yükle

Etiket alanı

- Komut satırının ilk bilgisidir ve sembolik isimlerden oluşur.
- Program içerisindeki belirli işlevlerin gerçekleştiği bölümlerin başlangıcını göstermek amacı ile kullanılır.
- Program içerisinde istenilen kısma kolaylıkla dallanılmasını sağlar.
- Etiket ismi olarak mikroişlemci komut setinde tanımlı olan bir komut ismi verilemez.
- Etiket bir harf ile başlamak zorundadır.

Program Formatı

Komut

- Mnemonic tabanlı komut seti içerisinde mikroişlemcinin belirli bir işi yapmasını sağlayan tanımlanmış kısaltmalardır.
- Komut alanına etiketten sonra 1 boşluk ya da sekme (tab) ile girilir.

İşlenen (operand)

- Bu alan, işlemciye işlenecek veriyi ya da verinin nerede olduğunu gösterir.
- Tek başına bir anlam ifade etmez.
- Genelde komutun etki edeceği **hedef ve kaynak** bilgisini içerir. Hedef ve kaynak bilgisi birbirinden virgül (,) ile ayrılır.

Program Formatı

Assembly dilinde sayı sistemlerinin kullanımı

Ön Takı	Son Takı	Anlamı	Örnek
(Boşluk)	D	Onlu sayı (decimal)*	55 – 55D
%	B	İkili sayı (binary)	%01010101 – 01010101B
@	O	Sekizli sayı (octal)	@33 – 33O
$	H	Onaltılık sayı (hexadecimal)	$FB – FBH

Açıklama Satırı

- Assembly dili (;) ile başlayan satırları açıklama satırı olarak kabul eder.
- Bu satırları yorumlamaz ve makine kodu üretmez.
- Yazılan uygulamanın anlaşılırlığını arttırır

Yönergeler

ORG

- Kod bellek içerisinde programın başlangıç adresini belirtmek için kullanılan adres konumlandırma talimatıdır.
- ORG ‘Adres’ şeklinde kullanılır.
- Bir program içerisinde birden fazla ORG komutu kullanılabilir.

Talimat	**Açıklama**
`ORG 0000h`	`;program 0000h adresinden başlasın PC=0000`
`ORG 0030h`	`;program 0030h adresinden başlasın PC=0030`

Yönergeler

END

- Programın bitiğini gösteren talimattır.

DB (Define Bayt)

- Kod bellek içerisinde sayı ve kelime (string) dizilerinin tanımlanmasını sağlar.

İsim	DB	ifadeler	Açıklama
Max_sayi	DB	255	;tek bir değişken ;tanımlanması
Tablo	DB	0, 5, 4, 3, -10	;dizi olarak tanımlama
Yaz	DB	‘8051 ogreniyorum’	;string olarak tanımlama

Yönergeler

EQU

- EQU (Equal = eşittir) bir sayısal değerin istenilen sembol adına atanması işlemini gerçekleştirir

- Bu tanımlama program içerisinde bir ifadenin ya da değerin çok fazla tekrar edildiğinde programın anlaşılırlığını arttırmak için kullanılır.

İsim	Talimat	Değer	Açıklama
Pi	EQU	3.14	;sabit değer tanımlama
Bilgi	EQU	55h	;55h adresindeki veriyi bilgi değişkenine ata

Komut Türleri

- 8051'de de kullanılan komutlar 8-bit işkoduna sahiptir.
- 8-bit işkodu 2^8=256 farklı komuta imkan tanır ve 8051'de toplam 255 komut tanımlıdır.
- 8051 komut kümesi 1, 2 ya da 3 bayt uzunluğunda komutlardan meydana gelmektedir.
- Komut kümesini oluşturan 255 komutun 139'u 1 bayt, 92'si 2 bayt ve 24'ü 3 bayttır.
- 8051 komut kümesi beş ana başlık altında incelenebilir:
  - Aritmetik komutlar
  - Mantıksal komutlar
  - Veri transfer komutları
  - Bit işlem komutları
  - Program dallanma komutları

Aritmetik Komutlar

- Toplama, çıkarma, artırma, azaltma, çarpma, bölme ve onluk tabana uyarlama komutlarından oluşmaktadır.

	Komut		Açıklama	Bayrak	Bayt
Toplama	ADD	A, Rn	Rn saklayıcı değerini akümülatöre ekle	C,OV,AC	1
	ADD	A, adres	Adresteki bilgiyi Aküye ekle	C,OV,AC	2
	ADD	A, @Ri	Saklayıcının gösterdiği adresteki bilgiyi aküye	C,OV,AC	1
	ADD	A, #bilgi	Doğrudan bilgiyi aküye ekle	C,OV,AC	2
Eldeli Toplama	ADDC	A, Rn	Akümülatör ile saklayıcı değerini elde ile topla	C,OV,AC	1
	ADDC	A, adres	Elde ile aküye adresteki bilgiyi ekle	C,OV,AC	2
	ADDC	A, @Ri	Elde ile saklayıcının gösterdiği adresteki bilgiyi aküye ekle	C,OV,AC	1
	ADDC	A, #bilgi	Elde ile doğrudan bilgiyi aküye ekle	C,OV,AC	2
Çıkarma	SUBB	A, Rn	Borç ile Aküden saklayıcının değerini çıkart	C,OV,AC	1
	SUBB	A, adres	Borç ile aküden adresteki bilgiyi çıkart	C,OV,AC	2
	SUBB	A, @Ri	Borç ile saklayıcının gösterdiği adresteki bilgiyi aküden çıkart	C,OV,AC	1
	SUBB	A, #bilgi	Borç ile Aküden bilgiyi çıkart	C,OV,AC	2

Aritmetik Komutlar

Arttırma	INC	A	Akümülatörü 1 arttır	–	1
	INC	Rn	Saklayıcıyı 1 arttır	–	1
	INC	adres	Adresteki bilgiyi 1 arttır	–	2
	INC	@Ri	Ri saklayıcının gösterdiği adresteki bilgiyi 1 arttır	–	1
	INC	DPTR	DPTR saklayıcısını 1 arttır	–	1
Azaltma	DEC	A	Akümülatörü 1 azalt	–	1
	DEC	Rn	Saklayıcıyı 1 azalt	–	1
	DEC	adres	Adresteki bilgiyi 1 azalt	–	2
	DEC	@Ri	Ri saklayıcısının gösterdiği adresteki bilgiyi 1	–	1
Çarpma	MUL	AB	A ve B saklayıcılarının içeriklerini çarp. Çarpım sonucunda yüksek değerlikli bayt B saklayıcısına, düşük değerlikli bayt ise Akü'ye	C,OV	1
Bölme	DIV	AB	A'yı B'ye böl. İşlem sonucunda Bölüm Akü'ye, Kalan B saklayıcısına yüklenir.	C,OV	1
	DA	A	Akümülatörü onluk tabana ayarla	C	1

Mantıksal Komutlar

- Lojik işlem komutları VE, VEYA, Özel VEYA, sola ve sağa döndürme komutları ile akünün 4'lüklerinin (nibble) yerini değiştirme komutundan meydana gelmektedir.

	Komut		Açıklama	Bayrak	Bayt
VE işlemi	ANL	A, Rn	Rn saklayıcısı ile aküyü VE işlemine tabi tut	–	1
	ANL	A, adres	Adresteki bilgi ile aküyü VE işlemine tabi tut	–	2
	ANL	A, @Ri	Saklayıcının gösterdiği adresteki bilgi ile aküyü lojik VE işlemine tabi tut	–	1
	ANL	A, #bilgi	Doğrudan bilgi ile aküyü VE işlemine tabi tut	–	2
	ANL	adres, A	Akü ile adresteki bilgiyi VE işlemine tabi tut	–	2
	ANL	adres, #bilgi	Bilgi ile adresteki veriyi VE işlemine tabi tut	–	3
VEYA işlemi	ORL	A, Rn	Rn ile akümülatörü VEYA işlemine tabi tut	–	1
	ORL	A, adres	Adresteki bilgi ile aküyü VEYA işlemine tabi tut	–	2
	ORL	A, @Ri	Saklayıcının gösterdiği adresteki bilgi ile aküyü VEYA işlemine tabi tut	–	1
	ORL	A, #bilgi	Doğrudan bilgi ile aküyü VEYA işlemine tabi tut	–	2
	ORL	adres, A	Akü ile adresteki bilgiyi VEYA işlemine tabi tut	–	2
	ORL	adres, #bilgi	Bilgi ile adresteki veriyi VEYA işlemine tabi tut	–	3

Mantıksal Komutlar

Özel VEYA işlemi	XRL	A, Rn	Rn ile aküyü Özel VEYA işlemine tabi tut	–	1
	XRL	A, adres	Adresteki bilgi ile aküyü Özel VEYA işlemine tabi tut	–	2
	XRL	A, @Ri	Saklayıcının gösterdiği adresteki bilgi ile aküyü Özel VEYA işlemine tabi tut	–	1
	XRL	A, #bilgi	Doğrudan bilgi ile aküyü Özel VEYA işlemine tabi tut	–	2
	XRL	adres, A	Akü ile adresteki bilgiyi Özel VEYA işlemine tabi tut	–	2
	XRL	adres, #bilgi	Bilgi ile adresteki veriyi Özel VEYA işlemine tabi tut	–	3
Döndürme İşlemi	RL	A	Akümülatörü 1 bit sola döndür	–	1
	RR	A	Akümülatörü 1 bit sağa döndür	–	1
	RLC	A	Akümülatörü elde üzerinden 1 bit sola döndür	C	1
	RRC	A	Akümülatörü elde üzerinden 1 bit sağa döndür	C	1
	SWAP	A	Akümülatörün ilk dört biti ile son dört bitini yer değiştir	–	1

Veri Transfer Komutları

- Bellekten veya G/Ç portlarından saklayıcılara ya da saklayıcılardan belleğe veri taşımak için kullanılırlar.

- 8051'de kullanılan veri transfer komutları 3 başlık altında toplanabilir.

  - Dahili veri belleğine erişmek için kullanılanlar
  - Harici veri belleğine erişmek için kullanılanlar
  - Program belleğine ya da bakış tablolarına (look-up table) erişmek için kullanılanlar

Dahili Veri Belleği Transfer Komutları

Komut		Açıklama	Bayt
MOV	A, Rn	Rn saklayıcısındaki değeri akümülatöre yükle	1
MOV	A, adres	Adresteki bilgiyi aküye yükle	2
MOV	A, @Ri	Ri'nin gösterdiği adresteki bilgiyi aküye yükle	1
MOV	A, #bilgi	Doğrudan *bilgi* verisini aküye yükle	2
MOV	Rn, A	Akümülatörü Rn saklayıcısına yükle	1
MOV	Rn, adres	Adresteki bilgiyi Rn saklayıcısına yükle	2
MOV	Rn, #bilgi	Doğrudan *bilgi* verisini Rn saklayıcısına yükle	2
MOV	adres, A	Akümülatördeki bilgiyi adrese yükle	2
MOV	adres, Rn	Rn saklayıcısının içeriğini adrese yükle	2
MOV	adres1, adres2	adres 2'deki bilgiyi adres 1'e yükle	3
MOV	adres, @Ri	Ri'nin gösterdiği adresteki bilgiyi adrese yükle	2
MOV	adres, #bilgi	Doğrudan *bilgi* verisini adrese yükle	3
MOV	@Ri, A	Akümülatörü Ri'nin gösterdiği adrese yükle	1
MOV	@Ri, adres	Adresteki bilgiyi Ri'nin gösterdiği adrese yükle	2
MOV	@Ri, #bilgi	Doğrudan *bilgi* verisini Ri'nin gösterdiği adrese yükle	2
MOV	DPTR, #bilgi16	16 bitlik *bilgi* verisini DPTR saklayıcısına yükle	3
PUSH	Adres	Adresteki bilgiyi yığına at	2
POP	Adres	Yığındaki bilgiyi adrese at	2
XCH	A, Rn	Rn ve akünün içeriklerini değiştir	1
XCH	A, adres	Adresteki bilgi ile akünün içeriğini değiştir	2
XCH	A, @Ri	Ri'nin gösterdiği adres ve akünün içeriklerini değiştir	1
XCHD	A, @Ri	Ri'nin gösterdiği adres ile akünün içeriklerinin ilk dört bitini değiştir	1

Harici Veri Belleği Transfer Komutları

Komut	Açıklama	Bayt
MOVX A, @Ri	Ri saklayıcısının gösterdiği harici RAM adresindeki veriyi akümülatöre yükle	1
MOVX @Ri, A	Aküyü Ri'nin gösterdiği harici RAM adresine yükle	1
MOVX A, @DPTR	DPTR'nin gösterdiği harici RAM (16 bitlik adres) adresindeki bilgiyi aküye yükle	1
MOVX @DPTR, A	Aküyü DPTR'nin gösterdiği harici RAM adresine yükle	1

Program Belleği Transfer Komutları

Komut	Açıklama	Bayt
MOVC A, @A+DPTR	A+DPTR'nin gösterdiği harici ROM (16 bitlik adres) adresindeki veriyi Akümülatöre yükle	1
MOVC A, @A+PC	A+PC'nin gösterdiği harici ROM (16 bitlik adres) adresindeki veriyi Akümülatöre yükle	1

Bit-İşlem Komutları

Komut		Açıklama	Bayrak	Bayt
CLR	A	Akümülatörü temizle	–	1
CPL	A	Akümülatörü tersle	–	1
CLR	C	Eldeyi sıfırla	C	1
CPL	C	Eldeyi tersle	C	1
SETB	C	Eldeyi birle (C = 1)	C	1
SETB	bit	Bit adreslenebilir RAM'deki bir bitlik veriyi birle	–	2
CLR	bit	Bit adreslenebilir RAM'deki bir bitlik veriyi birle	–	2
CPL	bit	Bit adreslenebilir RAM'deki bir bitlik veriyi tersle	–	2
MOV	C, bit	Bir bitlik adresteki veriyi elde bayrağına yükle	C	2
MOV	bit, C	Eldeyi bir bitlik adrese yükle	C	2
ANL	C, bit	Elde ile bir bitlik veriyi VE işlemine tabi tut	C	2
ORL	C, bit	Elde ile bir bitlik veriyi VEYA işlemine tabi tut	C	2

Program Dallanma Komutları

- Program Dallanma komutları
  - Şartsız dallanma,
  - Şartlı dallanma
  - Alt program çağırma ve alt programdan dönme komutları

Şartsız Dallanma Komutları

Komut		Açıklama	Bayt
SJMP	Adres	Kısa dallanma (adrese dallan)	2
AJMP	adres11	Mutlak adresleme yönteminde kullanılır, 11 bitlik adres (2 KB) alanı içerisinde bir dallanma sağlar	2
LJMP	adres16	Uzun dallanma, 16 bitlik adres alanı içerisinde bir atlama sağlar	3
JMP	@A+DPTR	A+DPTR'nin gösterdiği adrese dallan	3
NOP		1 makine çevrim boyu işlem yapma	1

Program Dallanma Komutları

Şartlı Dallanma Komutları

Komut		Açıklama	Bayrak	Bayt
JC	Adres	Eğer C = 1 ise adrese dallan	–	2
JNC	Adres	Eğer C = 0 ise adrese dallan	–	2
JB	bit, adres	Eğer bit = 1 ise adrese dallan	–	3
JNB	bit, adres	Eğer bit = 0 ise adrese dallan	–	3
JBC	bit, adres	Eğer bit = 1 ise adrese dallan sonra biti sıfırla (bit = 0)	–	3
JZ	Adres	Eğer akümülatör sıfır (A = 0) ise adrese dallan	–	2
JNZ	Adres	Eğer A = 0 değil ise adrese dallan	–	2
DJNZ	Rn, adres	Rn'i bir azalt ve Rn sıfır değilse adrese dallan	–	2
DJNZ	adres1, adres2	Adres1'deki veriyi 1 azalt, eğer sıfır değilse adres2'ye dallan	–	3
CJNE	A, adres1, adres2	Akü ve adres1'deki veriyi karşılaştır, eşit değilse adres2'ye dallan	C	3
CJNE	A, #bilgi, adres	Akü ve *bilgiyi* karşılaştır, eşit değilse adrese dallan	C	3
CJNE	Rn, #bilgi, adres	Rn ve *bilgiyi* karşılaştır, eşit değilse adrese dallan	C	3
CJNE	@Ri, #bilgi, adres	Ri'nin gösterdiği adresteki veri ile *bilgiyi* karşılaştır, eşit değilse adrese dallan	C	3

Program Dallanma Komutları

Alt program çağırma ve alt programdan dönme komutları

Komut	Açıklama	Bayt
ACALL adres11	Adres11 etiketli alt programı çağır	*2*
LCALL adres16	Adres16 etiketli alt programı çağır	*3*
RET	Alt programdan kaldığın yere dön	*1*
RETI	Kesme alt programından kaldığın yere dön	*1*

Bölüm Soruları

1. Assembly dilinin günümüzde hala belirli alanlarda kullanılmasını gerekçelendiriniz..
2. Assembly dilinde yazılan bir program ile C gibi yüksek seviyeli dilde yazılan programların arlarındaki temel farklar nelerdir.
3. Assembler (çevirici) ile compiler (derleyicinin) yaptıkları iş açısından temel farklılıkları neler olabilir?
4. Yönerge ile komut arasındaki farkı açıklayınız.
5. Assembly program yazdığınız zaman, hedef mikroişlemci/Mikrodenetleyici donanımına ve mimarisine daha hakim olduğunuzu hissedebiliyor musunuz? Örnekler veriniz…
6. Mikroişlemcilerde bit temelli komutların olmayışının sebeplerini araştırınız