Mikro - Final (4. OCAK 2016)

1) PC'den gönderilen 16 adet ASCII kodlu klavye karakter dizisini seri portdan alıp (9600 baud, 8 bit, 1 start, 1 stop), bu karakterleri 100H adresinden itibaren dahili hafızaya kaydeden ve arka arkaya gelen iki karakter aynı ise işlemi sonlandıran assembler programı yazınız.

```
        MOV TMOD,#20H  ---> Timer 1, Mod 2
        MOV TH1,#-3    ---> 9600 Baud Rate
        MOV SCON,#50H  ---> 1 start, 1 stop, 8 bit
        SETB TR1       ---> Timer aktif
        MOV R1,#0
        MOV @DPTR,100H
BASLA
        MOV A,SBUF
BEKLE
        JNB TI,BEKLE
DELAY
        MOVX @DPTR,A
        INC @DPTR
        CJNE A,R1,DEVAM
        SJMP BITIR
DEVAM
        MOV R1,A
        SJMP BASLA
BITIR
        END
```

2) Dış belleklerde (Harici RAM bölgesi) 2000H adresinden itibaren 1KB'lik bölgeyi temizleyen (0 yükleyen) assembler programını yazınız.

```
        $MOD842
        CSEG
        ORG 0000H
BASLA:
        MOV DPTR,#2000H
        MOV R0,#128     -> 128 . 8 = 1024 = 1KB
        MOV R1,#8
        MOV A,#00
DONGU0
        MOV R1,#8
DONGU1
        MOV @DPTR,A
        INC DPTR
        DJNZ R1,DONGU1
        DJNZ R0,DONGU0
        END
```

(3) Bir sensörün çıkış işaretini T1/C1 girişinden sayarak (counter 1 kullanarak) sayan (2) bir devre düşünün. Bu devrenin verdiği darbeleri sayar ve sayı 20.000 olunca P1.7 bitini set eden assembler programını yazınız.

```
#MOD 842
CSEG
ORG 0000H
MOV TMOD, #50H
MOV TH1 , #B1
MOV TL1 , #E0
SETB TR1
SETB P3.5
JNB TF1 , $
CLR TR1
CLR TF1
SETB P1.7
END
```

GATE	C/T	M1	M0	GATE	C/T	M1	M0

65536 − 20000 = 45536 (B1E0)

45536 | 16
2846 | 16
177 | 16
11 | 1

(3) (16) (1) (14) → B1E0

(44) Aşağıdaki programda bir seri haberleşme işlemi yapılmıştır. 9600 Baud hızında 8 bit data, 1 start, 1 stop bit olacak şekilde yapılmış bu seri haberleşme programının bütününde neyapılmak istendiğini yazarak nokta nokta olarak belirtilen eksik satırları tamamlayınız.

```
ORG 0000H
JMP MAIN
ORG 0100H
MAIN:
    CALL BAUDR
    MOV A, #'A'
    CALL SEND
    MOV A, #'B'
    CALL SEND
    MOV A, #'C'
    CALL SEND
    JMP $
SEND:
    MOV SBUF, A
    JNB TI , $
    CLR TI
    RET
BAUDR:
    MOV T3CON, #82H
    MOV T3ED, #20H
    MOV SCON, #52H
    RET
END
```

MOV TMOD #20H
MOV TH1 , #-3
MOV SCON , #50H
de yazılabilir.

Mikrodenetleyiciler ve Programlama

(6 Ocak 2015)
BÜT

① Akümülatöre 10 defa 3 sayısını bir döngü isinde ekleyen programı yazınız.

```
        MOV  R0,#10
        MOV  R1,#03
DONGU:
        ADD  A,R1
        DJNZ R0,DONGU
        END
```

② Akümülatöre 55H sayısını yükleyip daha sonra 700 defa akümülatörün tümleyenini (Complement) alan programı yazınız.

```
         MOV  R0,#70
         MOV  R1,#10
         MOV  A,#55H
DONGU1 : MOV  R1,#10
DONGU-2: CPL  A
         DJNZ R1,DONGU2
         DJNZ R0,DONGU1
         END
```

③ Giriş bilgilerini P1 portundan okuyup bu bilgileri seri porttan gönderen (Transmit) programını yazınız. (8 bit data, 1 stop bit ve 9600 baudrate)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI	SCON
0	1	0	1	0	0	0	0	

5 0 = 50H

01 → Mod 1 = 8 bit data

GATE	C/T	M1	M0	GATE	C/T	M1	M0	TMOD
0	0	1	0	0	0	0	0	

Timer1 / Timer0 → = 20H

Timer 1, Mod 2 = 8 bit

(NOT SCON Mod1 = TCON Mod2 Timer1)

9600 baudrate = -3

```
       MOV  SCON,#50H
       MOV  TMOD,#20H  → (Timer1, Mode2 (8 bit autoreload)
       MOV  TH1,#-3    → (TH1=-3, the high byte)
       SETB TR1
       MOV  P1,#0FFH
DONGU: MOV  A,P1
       MOV  SBUF,A
       JNB  TI,$
       CLR  TI
       JMP  DONGU
       END
```

TMOD:

Gate	C/T	M1	M0	Gate	C/T	M1	M2
TIMER1				TIMER2			

(yukarı / aşağı sayıcı)

Timer or counter (0=T, 1=C)

M1 M0	MODE	Operation Mode
00	0	13 bit timer
01	1	16 bit timer
10	2	8 bit autoreload
11	3	split timer

→ SETB TR1 → Timer1 aktif eder

④ Timer kullanmadan %50 duty cycle oranında kare dalga üreten ve bu işareti P1.0 ucundan çıkışa veren programı yazınız.

```
HERE:   CPL   P1.0
        LCALL DELAY
        SJMP  HERE
DELAY:  MOV   R0,#0FFH
        DJNZ  R0,$
        RET
```

II. Yol

```
HERE:   SETB  P1.0
        LCALL DELAY
        CLR   P1.0
        LCALL DELAY
DELAY:  MOV   R0,#0FFH
        DJNZ  R0,$
        RET
```

FINAL

① 50H adresinden itibaren 4 adet byte (8 bit) datayı 60H adresinden itibaren 4 adet byte (8 bit) data ile toplayıp sonuçları 80H adresinden itibaren hafızaya ve LCD'ye yazan programı yazınız. (Toplam 8 bit varsayınız)

```
       $MOD842
       CSEG
       ORG   0100H
MAIN:  MOV   A,#01H
       ACALL KOMUTYAZ
       ACALL DELAY
       MOV   A,#0EH
       ACALL KOMUTYAZ
       ACALL DELAY
       MOV   A,#30H
       ACALL KOMUTYAZ
       ACALL DELAY
       MOV   R0,#50H
       MOV   R1,#60H
       MOV   R2,#80H
       MOV   R3,#04
TOPLA: MOV   A,@R0
       ADD   A,@R1
       MOV   @R2,A
       ACALL BILGISAYAR
       INC   R0
       INC   R1
       INC   R2
       DJNZ  R3,TOPLA
       END

→BILGISAYAR: MOV   P0,A
             SETB  P2.0
             CLR   P2.1
             SETB  P2.2
             ACALL DELAY
             CLR   P2.2
             RET

→KOMUTYAZ:   MOV   P0,A
             CLR   P2.0
             CLR   P2.1
             SETB  P2.2
             ACALL DELAY
             CLR   P2.2
             RET
```

TERMO ISI

TARİH/DATE

BÜTÜNLEME

1) Akümülatöre 10 dan 3 sayısını bir döngü içerisinde ekleyen program

```
        MOV   R0, #10
        MOV   R1, #03
DONGU:
        ADD   A, R1

        DJNZ  R0, DONGU
        END
```

2) Akümülatöre 55H sayısını yükleyip daha sonra 200 defa akümülatörün tümleyenini alan program

```
        MOV   A , #55H
        MOV   R0 , #200

DELAY_0: MOV  R1 , #10
DELAY_1: CPL  A

        DJNZ  R1 , DELAY_1
        DJNZ  R0 , DELAY_0
        END
```

3) Giriş bilgilerini P1 portundan okuyup bu bilgileri seri porttan gönderen (transmitter) programını yazınız (8 bit data, 1 stop bit ve 9600 baudrate)

```
        MOV   SCON , #50H
        MOV   TMOD , #20H
        MOV   TH1 , #-3
        SETB  TR1
        MOV   P1 , #0FFH
DONGU:
        MOV   A , P1
        MOV   SBUF , A
        JNB   TI , $
        CLR   TI
        JMP   DONGU
        END
```

SCON (veri almak için 1 olmalı)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
0	1	0	1	0	0	0	0

0 1 → Mod 1 8 bit

TMOD

T1				T0			
GATE	C/T	M1	M0	GATE	C/T	M1	M0
0	0	1	0	0	0	0	0

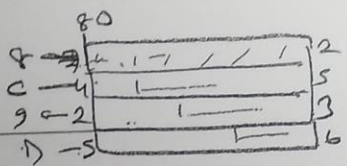
Timer 1

1 0 → mod 2

9600 baudrate => TH1 = -3

(SCON Mod 1 iken Timer 1 Mod 2 iste)

Pt	Sa	Ça	Pe	Cu	Ct	Pz



...../...../.....

(CO → 2. satır
80 → 3. satır) 10

Timer 1 kullanılarak %50 dolu-boş oranlı bir kare dalga üretiniz. Kare dalga P1.0 bacağından dışarıya verilecek ve kare dalganın High olduğu durumda LCD ekranda 2 satırın başında ilgili 1, kare dalganın low olduğu durumda " " 3. " " " 0, yazısı görünecek.

```
$MOD52
CSEG
ORG     0000H

MOV     A, #01H
ACALL   KOMUT
ACALL   DELAY
MOV     A, #0EH
ACALL   KOMUT
ACALL   DELAY
MOV     TMOD, #10H  → Timer1
SETB    P1.0
HERE:   MOV     TH1, #0FFH
        MOV     TL1, #0EEH
        CPL     P1.0
        SETB    TR1
        JNB     TF1, $
        CLR     TR1
        CLR     TF1
        MOV     A, P1.0
        CJNE    A, #00H, LCD1
        JMP     LCD2

DELAY:

KOMUT

BILGI

LCD1:   MOV     A, #80H
        ACALL   KOMUT
        ACALL   DELAY
        MOV     A, #'0'
        ACALL   BILGI
        ACALL   DELAY
        JMP     HERE
LCD2:   MOV     A, #0C0H
        ACALL   KOMUT
        ACALL   DELAY
        MOV     A, #'1'
        ACALL   BILGI
        ACALL   DELAY
        JMP     HERE
```

Pt | Sa | Ça | Pe | Cu | Ct | Pz

..../..../(5)

50H adresinden itibaren 4 adet byte (8 bitlik) data'yı, 60H adresinden itibaren 4 adet byte (8 bitlik) data ile toplayıp, sonuçları 80h adresinden itibaren hafızaya ve LCD'ye yazan program (toplam sonucun 8 bit olacağı varsayılır)

```
$MOD852
CSEG
ORG     0100H
MAIN:   MOV     A, #01H
        ACALL   KOMUTYAZ
        ACALL   DELAY
        MOV     A, #0EH
        ACALL   KOMUTYAZ
        ACALL   DELAY
        MOV     A, #38H
        ACALL   KOMUTYAZ
        ACALL   DELAY

        MOV     R0, #50H
        MOV     R1, #60H
        MOV     R2, #80H
        MOV     R7, #04
TOPLA:
        MOV     A, @R0
        ADD     A, @R1
        MOV     @R2, A
        ACALL   BILGIYAZ
        INC     R0
        INC     R1
        INC     R2
        DJNZ    R7, TOPLA
        END
BILGIYAZ:
        MOV     P0, A
        SETB    P2.0
        CLR     P2.1
        SETB    P2.2
        ACALL   DELAY
        CLR     P2.2
        RET

KOMUTYAZ:
        MOV     P0, A
        CLR     P2.0
        CLR     P2.1
        SETB    P2.2
        ACALL   DELAY
        CLR     P2.2
        RET
```

maximum

P0 portunun her bacağına bir ledin anot ucunu bağladığımızı ve ledin katot ucunu da seri bir direnç ekleyerek toprağa bağladığımızı düşünün. Aynı anda sadece 2 led yanması şartıyla sürekli olarak ledleri ileri geri olarak yakıp söndüren ve her seferinde ledlerin gösterdiği sayısal değeri decimal olarak LCD'ye yazan program

```
        $MOD842
        CSEG
        ORG     0100H
        MOV     A, #01H
        ACALL   KOMUT
        ACALL   DELAY
        MOV     A, #0EH
        ACALL   KOMUT
        ACALL   DELAY
        MOV     A, #30H
        ACALL   KOMUT
        ACALL   DELAY
MAIN:   MOV     DPTR, #TABLE

        MOV     A, #00H
        MOV     P2, A
        MOV     R7, #6
AGAIN:  CLR     A
        MOVC    A, @A+DPTR
        MOV     P2, A
        ACALL   BILGI
        INC     DPTR
        ACALL   DELAY
        DJNZ    R7, AGAIN
        SJMP    MAIN
DELAY:  MOV     R0, #5
DLY1:   MOV     R1, #255
DLY2:   MOV     R2, #255
        DJNZ    R2, $
        DJNZ    R1, DLY2
        DJNZ    R0, DLY1
        RET

        ORG     200H
TABLE   DB      81h, 42h, 24, 18, 24, 42, 81
        END

BILGI:  MOV     P0, A
        SETB    P2.0        1   RS
        CLR     P2.1        0   R/W
        SETB    P2.2        1   E
        ACALL   DELAY
        CLR     P2.2        0
        RET

KOMUT:  MOV     P0, A
        CLR     P2.0        0
        CLR     P2.1        0
        SETB    P2.2        1
        ACALL   DELAY
        CLR     P2.2        0
        RET
```

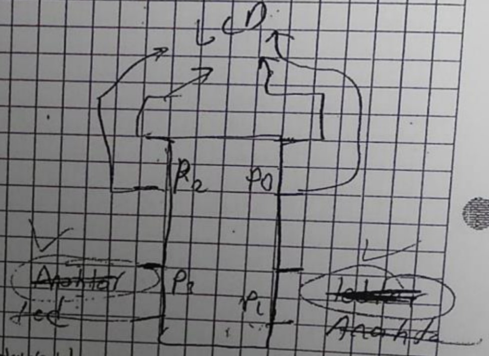
Pt | Sa | Ça | Pe | Cu | Ct | Pz

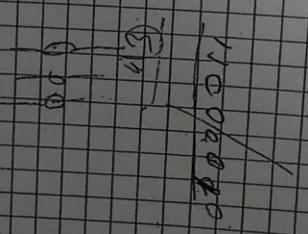
....../....../ 7

P1 portunun uclarına bağlı anahtarlardan okunan değeri (anahtar kapalıyken 0 okunur) P0 portuna bağlı ledlere yazan ve her seferinde ledlerin gösterdiği sayısal değeri decimal olarak LCD'ye yazan

```
        $MOD842
        CSEG
        ORG     0100H
        MOV     A, #01H
        ACALL   KOMUT
        ACALL   DELAY
        MOV     A, #0EH
        ACALL   KOMUT
        ACALL   DELAY
        MOV     A, #70H
        ACALL   KOMUT
        ACALL   DELAY
AGAIN:  CLR     P3
        MOV     A, P3     → (P3 8bit bilgi okundu)
        MOV     P1, A     →
        ACALL   YENI      — LCD P0 bağlı
        INC     DPTR
        ACALL   DELAY
        SJMP    AGAIN
DELAY:  MOV     R0, #5
DLY1:   MOV     R1, #255
DLY2:   MOV     R2, #0FFH
        DJNZ    R2, $
        DJNZ    R1, DLY2
        DJNZ    R0, DLY1
        RET
KOMUT:
        MOV     P0, A
        CLR     P2.0
        CLR     P2.1
        SETB    P2.2
        ACALL   DELAY
        CLR     P2.2
        RET
```

MOV R0, A





BILGI:

```
MOV     P0, A
SETB    P2.0
CLR     P2.1
SETB    P2.2
ACALL   DELAY
CLR     P2.2
RET
```

maximum

Pt | Sa | Ça | Pe | Cu | Ct | Pz

_____ / _____ / 8

Sınavsız geçiş hakkı veilmiş 5 sayıdan en büyük sayıyı bulan ve sesli şekilde bu sayıyı PC'ye gönderen ve aynı değeri LCD'ye yazan assembler programı (seri port ögesi 8600 baudrate, 8 bit data, 1 stop bit)

```
$MOD812
ORG 00H
MAIN:   MOV   T2CON, #08H
        MOV   TH2, #0FDH
        MOV   SCON, #52H
        MOV   A, #01H
        ACALL KOMUT
        ACALL DELAY
        MOV   A, #0EH
        ACALL KOMUT
        ACALL DELAY
        MOV   A, #80H
        ACALL KOMUT
        ACALL DELAY
EN_B_D: E_B   DATA 31H
        SAYAC DATA 32H
        MOV   R0, #300H
        MOV   R1, #300H
        MOV   SAYAC, #5
CIKAR:  MOV   A, @R0
        SUBB  A, @R1
        JC    DEGIS
DEGISME: INC  R1
        MOV   E_B, R0
        DJNZ  SAYAC, CIKAR
        RET
DEGIS:  MOV   A, R1
        MOV   R0, A
        INC   R1
        MOV   E_B, R0
        DJNZ  SAYAC, CIKAR
        RET
        MOV   A, E_B
        MOV   SBUF, A
        JNB   TI, $
        CLR   TI
        ACALL BILGI

        ORG 300H
TABLE:  DB 3,5,1,4,6
        END

KOMUT   BILGI
```

300 | 3
301 | 5
302 | 1
303 | 4
304 | 6

Pt | Sa | Ça | Pe | Cu | Ct | Pz

....../....../9

Aşağıdaki programda ADC 3 ile yapılan tekli çevrimler ile bir analog işaret dijitale çevrildikten sonra 12 bit olarak çevrilmektedir. Analog işaret dijital işaretle 12 bit olarak çevrilmektedir (veriler adreslere dekil) önce sonra 40 Hex adresinden 7F Hex adresine kadar High byte'ı sonra low byte'ı olmak üzere hafızaya atılmaktadır. Her bir satırdaki komutları açıklayarak nokta nokta olarak belirtilen eksikleri tamamlayınız.

```
        $MOD842
        CSEG
        ORG     000H
        JMP     MAIN

        ORG     04BH
MAIN:   MOV     R0 , #40H
        MOV     ADCCON1 , #E0H
        MOV     ADCCON2 , #03H
TEKRAR:
        CJNE    R0 , #07FH , DEVAM
        END
DEVAM:  SETB    SCONV
        JNB     ADCI , $
        CLR     ADCI
        MOV     R1 , ADCDATAH
        ANL     01H , #0FH
        MOV     @R0 , R1
        INC     R0
        MOV     @R0 , ADCDATAL
        INC     R0
        JMP     TEKRAR
```

00 → R0
01 – R1
10 – R2

0 0 1 1 = 3H

0011 1101
0000 1111
0000 1101

R0 0011 1101
0F 0000 1111
ANL 0000 1101

TERMO ISI

TARİH/DATE

4) 8 bitlik R0... R2 registerlerinin ha bilme yükleyebilecek en büyük desimal sayı 255 old. göre; bu registerleri kullanıp 65025 (255x255) defalık bir gecikme (delay) sağlayan iç içe döngü tekniğini yazınız. Her satır ile ilgili açıklama yapınız

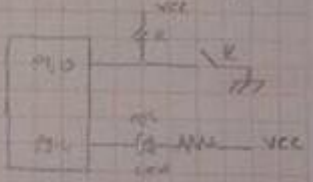
```
        MOV   R0 , # 255
        MOV   R1 , # 255
        MOV   R2 , # 10

DELAY_0: MOV  R1 , # 255
DELAY_1: MOV  R2 , # 10
DELAY_2

        DJNZ  R2, DELAY_2
        DJNZ  R1, DELAY_1
        DJNZ  R0, DELAY_0

        END
```

6) İlk durumda K anahtarı açık ve led sönük. K anahtarı kapatıldığında ledin yanması isteniyor



K açık P1.0 = 1 Led = 0 P1.1 = 1
K kapalı P1.0 = 0 Led = 1 P1.1 = 0

SORULAR

1-) 64 KB' lık hafıza haritasında (Memory Map) **8KB'lık EPROM**, **2000 Hex adresinden**, **4KB'lık ROM**, **6000 Hex adresinden** ve **4KB'lık RAM**, **8000 Hex adresinden** itibaren yerleştirilmiştir. Bu entegreler için CS (Chip Select) üretilmesini sağlayacak lojik devreleri oluşturunuz. **(Tüm entegreleri ve bağlantıları (Mikrodenetleyici, Ram, Eprom, Rom..) tam olarak çiziniz**. Adress Bus ($A_0 - A_{15}$), Data Bus ($D_0 - D_7$)

2-) P1 portundan 8 adet led aşağıdaki senaryoya bağlı kalınarak yakılmıştır. Devresini çizerek assembler programı oluşturunuz. **(Enerji verildiğinde port lojik 1 olur !!)**

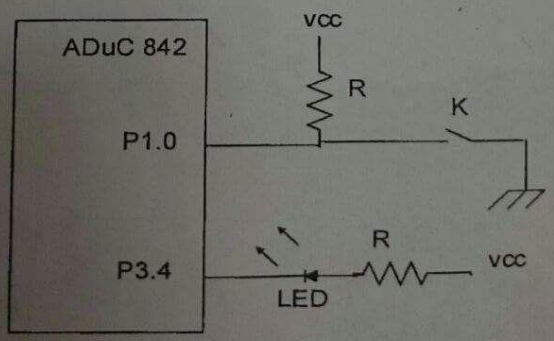
- Her zaman aynı anda sadece **2 led** yanacak şekilde; önce 2 baştan birer led yanacak, kısa bir gecikme olacak ve bu işlem her adımda merkeze , sonra merkezden dışa doğru ters işlemleri tekrar edecek. Bu işlem sürekli devam edecek.

3-) 50H adresinden itibaren 4 adet Byte (8 Bitlik) datayı, 60H adresinden itibaren 4 adet Byte (8 Bitlik) data ile toplayıp, sonuçları 80H adresinden itibaren hafızaya yazan asembler programı yazınız. **(Toplam sonucunun 8 Bit olacağını varsayınız.)**

4-) 8 Bitlik R0 ... R7 register'larının her birine yüklenebilecek en büyük desimal sayı 255 olduğuna göre; bu register'ları kullanıp 650250 (255 x 255 x 10) defalık bir gecikme (Delay) sağlayan programı iç-içe döngü kullanarak yazınız. Her satır ile ilgili açıklamayı yazınız.

5-) 30 Hex. adresinden, 7F Hex. Adresine kadar olan iç bellekteki dataları, 1000 Hex. Adresinden itibaren dış veri belleğine kopyalayan asembler programı yazınız.

6-) Aşağıdaki devrede ilk durumda K anahtarı açık ve Led sönük kabul ediliyor. K anahtarı kapatıldığı zaman ledin yanması isteniyor. Bunu sağlayacak programı yazınız.)



Süre 60 dakikadır. Dilediğiniz 3 soruyu cevaplayınız

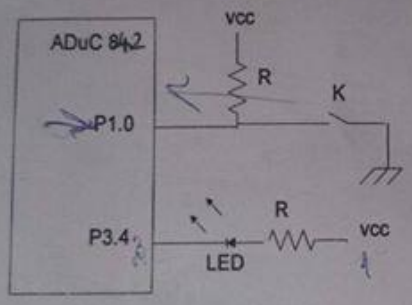
BAŞARILAR DİLERİM.

Yrd.Doç.Dr. C.Bülent FİDAN

```
MOV   TH1, #80H
SETB  ET1
SETB  EA          ; Bütün kesmelere (interrupt) izin ver
SETB  TR1
JNB   $
END
```

2) Aşağıdaki devrede ilk durumda K anahtarı açık ve Led sönük kabul ediliyor. K anahtarı kapatıldığı zaman ledin yanması, açıldığı zaman sönmesi isteniyor. Bunu sağlayacak programı yazınız.

CEVAP: Programı Buraya Yazınız




```
MAIN:  MOV  P1, #0FFH  ; P1 giriş port oldu
TEKRAR: MOV  C, P1.0
        JC   TEKRAR
YAK:   CLR  P3.4
       JMP  MAIN
```

3) Harici Kesme İsteği 0 (External Interrupt 0) INT0 'a bağlı bir tuş olduğunu düşününüz. Bu tuşa basıldıkça LCD ekranda tuşa basılma sayısını gösteren programı yazınız. **(INT 0 için alt program Hex. Adresi 03H 'tir ve INT 0 'a izin verme biti EX0 biti' dir.)**

Süre 70 dk.
Başarılar Dilerim,

Yrd. Doç. Dr. C.Bülent FİDAN

1 Kasım 2014

MİKRODENETLEYİCİLER ve PROGRAMLANMASI VİZE SINAVI

Süre: 80 dk.
Adı Soyadı: **Okul Numarası:**

SORULAR

1) Mikroişlemcili bir tasarımda, 64 KB' lık hafıza haritasında (Memory Map) **20KB'lık EPROM** (4000) Hex. adresinden, **12 KB'lık RAM** (C000) Hex. Adresinden ve 1 adet ADC **3 Byte' lık adres** (2000) Hex. Adresinden itibaren yerleştirilmiştir. Verilen şartlar ışığında bu tasarımı gerçekleyen devreyi temel lojik kapılar ve 74138, 74139... gibi kod çözücü entegreler kullanarak oluşturunuz ve **tüm entegrelerin (MikroD., Latch, Ram, Eprom, ADC.. gibi) bağlantılarını tam olarak gösteriniz**. Adress Bus ($A_0 - A_{15}$), Data Bus ($D_0 - D_7$)

2) Aşağıdaki programda Timer 0 'ı (T0) yeniden yüklemeli 8 bit modunda (Mod 2) çalıştırıp, kesme (Interrupt) üreterek microdenetleyicinin P0.1 ucundan Dolu-Boş oranı (Duty Cycle) %50 olan bir kare dalga verecek asembler programı yazılmıştır. **Herbir satırdaki komutları açıklayarak nokta nokta olarak belirtilen eksik satırları tamamlayınız.**

```
$MOD842                 ;AduC842 için ön tanımların bulunduğu register
CSEG                    ;Asembler Kod yazmaya başlanacağını belirler komut

ORG   0000 H            ;Kod yazmaya başlanacak adresi gösterir
JMP   MAIN              ;
ORG   000B H            ; TF0 İnterrupt (Kesme) isteği adresi
...                     ;
...                     ;
...                     ;
ORG   0100 H            ;
MAIN:
MOV   TMOD, #02H        ;
...                     ;
...                     ;
...                     ;
SETB  EA                ; Tüm kesmelere izin verilir
SETB  ET0               ; T0 Bayrağına (Flag'a) izin verilir
SETB  TR0               ; Timer 0 çalışmaya başlar
JMP   $                 ;
END                     ;
```

3) 30 Hex. adresinden, 7F Hex. Adresine kadar olan iç bellekteki dataları, 1000 Hex. Adresinden itibaren dış veri belleğine kopyalayan asembler programı yazınız.

4) 8 Bitlik R0 ... R7 register'larının her birine yüklenebilecek en büyük desimal sayı 255 olduğuna göre; bu register'ları kullanıp 650250 (255 x 255 x 10) defalık bir gecikme (Delay) sağlayan programı iç-içe döngü kullanarak yazınız. Her satır ile ilgili açıklamayı yazınız.

Başarılar Dilerim

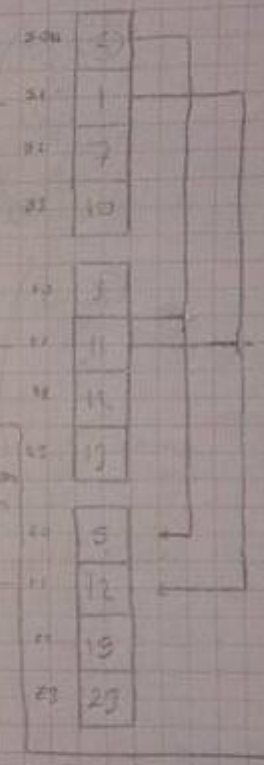
Yrd.Doç.Dr. Can Bülent Fidan

TERMO ISI

TARİH/DATE

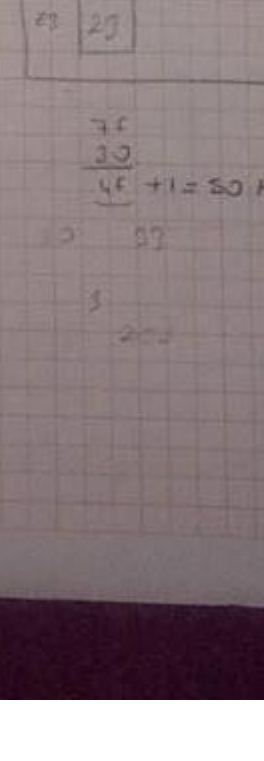
3) 30 H adresinden itibaren (4 adet) Byte (8 bitlik) dataya
50 H adresinden itibaren 4 adet Byte data ile toplayıp sonucu
80 H adresinden itibaren yazan assembler programını yazınız.

```
        MOV   R0, #30H
        MOV   R1, #50H
        MOV   R2, #80H
        MOV   R3, #04
DONGU:  MOV   A, @R0
        ADD   A, @R1
        MOV   @R2, A
        INC   R0
        INC   R1
        INC   R2
        DJNZ  R3, DONGU
        END
```



5) 30 Hex adresinden (7F H) adresine kadar sıralı bellekteki datalar 1000H adresinden itibaren dış ver belleğine kaydeden assembler yazın.

```
        MOV   R0, #30H
        MOV   R1, #(50H-1)
        MOV   DPTR, #1000H
DONGU:  INC   R0
        MOV   A, @R0
        MOVX  @DPTR, A
        INC   DPTR
        CJNE  R0, #7FH, DONGU
        DJNZ  R1, DONGU
        END
```



7F
30
4F +1 = 50 H

TERMO ISI

TARİH/DATE

VİZE

2) P1 portuna 8 adet led bağlanmıştır. Her zaman aynı anda 2 led yanacak önce 2 baştan birer tane yanacak, fiso bir geçilme olacak bu işlem önce merkeze sonra dışa doğru tekrarlanacak

1	0	0	0	0	0	0	1	=> 81H
0	1	0	0	0	0	1	0	=> 42H
0	0	1	0	0	1	0	0	=> 24H
0	0	0	1	1	0	0	0	=> 18H
0	0	1	0	0	1	0	0	=> 24H
0	1	0	0	0	0	1	0	= 42H
1	0	0	0	0	0	0	1	= 81H

```
        ORG   00H
MAIN:   MOV   R0, #06      ; döngü sayısı
        MOV   R1, #20H
DONGU:
        MOV   P1, @R1
        MOV   R2, #0FFH
      - DJNZ  R2, $
        INC   R1
        DJNZ  R0, DONGU

        JMP MAIN


        ORG   20H
MAY_DATA:
        DB  81H, 42H, 24H, 18H, 24H, 42H
```

R0	R1	P1
06	20H	81H

81H	20H
42H	21H
24	22H
18	23H
24	24H
42	25H