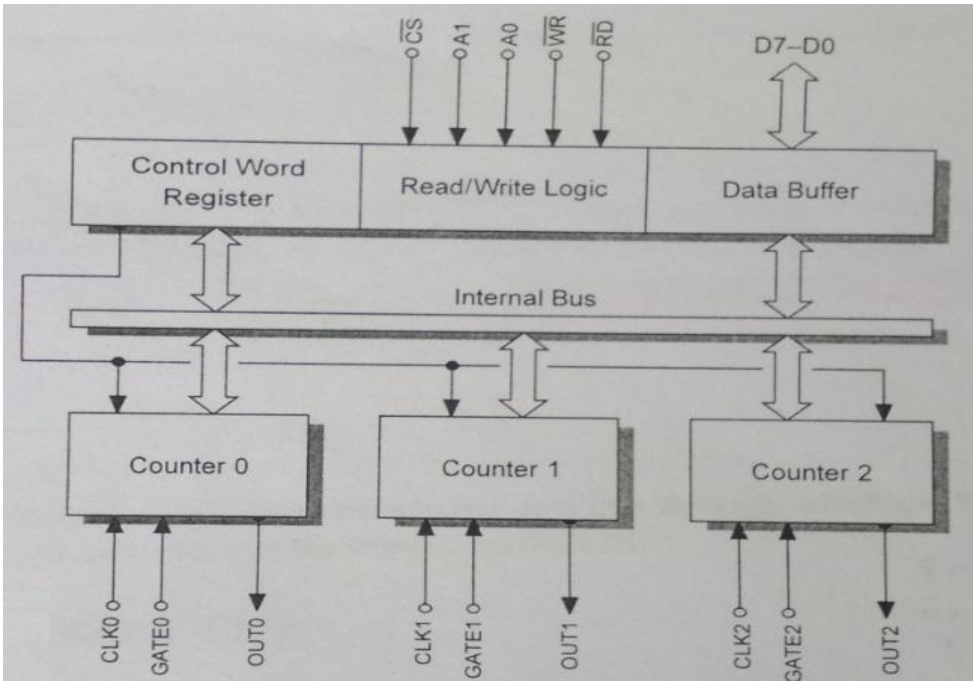
OLAYLARI ZAMANLAMA

İnsanların işlerini bir takvime ve zamana bağlı olarak yürütmesine benzer şekilde, bilgisayarlar da işlerini bir takvim ve zaman içinde yapar. Örneğin bir program koşulurken klavyeden yapılacak müdahaleleri anlayabilmek için belirli aralıklarla klavyenin yoklanması bir zamanlama işidir. Veya CMOS'ta üretilen gerçek-zamanlı saatten bilgisayarın sistem saatinin hangi aralıklarla güncelleneceği de yine bir zamanlama problemidir. Eski bilgisayarlarda dinamik belleklerin tazelenmesi (refresh) bu zamanlamaya verilecek başka bir örnektir.

Zamanlama işlemi donanım veya yazılımla yapılabilir. Yazılım yöntemi gecikme programı kullanımına dayanır ve bu yüzden CPU saat frekansının bir fonksiyonudur. Yani aynı gecikme programı farklı bilgisayarlarda farklı zamanlamalar doğurur. Bu farklılıkları ortadan kaldırmak ve bilgisayarın donanımından bağımsız zamanlamalar gerçeklemek için bilgisayarlarda özel donanımlar inşa edilmiştir. Bu özel donanımların başında da 1.193180 MHz'lik işaretin üretildiği saat devresi ve bu saatten farklı biçimlere sahip dikdörtgen dalga üretmeyi sağlayan 8254PIC Programlanabilen Aralık Zamanlayıcı gelir. Bu deneyin amacı 8254 PIC yongasını tanıtmak ve onu programlayarak çeşitli uygulamalar gerçeklemektir.

8254 PIC'in YAPISI ve ÇALIŞMASI

8254 PIC,CPU'dan bağımsız olarak, bir dış saat işaretinden programlanabilen zaman aralıkları üretir. Altı farklı çalışma moduna sahip olan PIC'in iç yapısı Şekil 1'de gösterilmiştir.

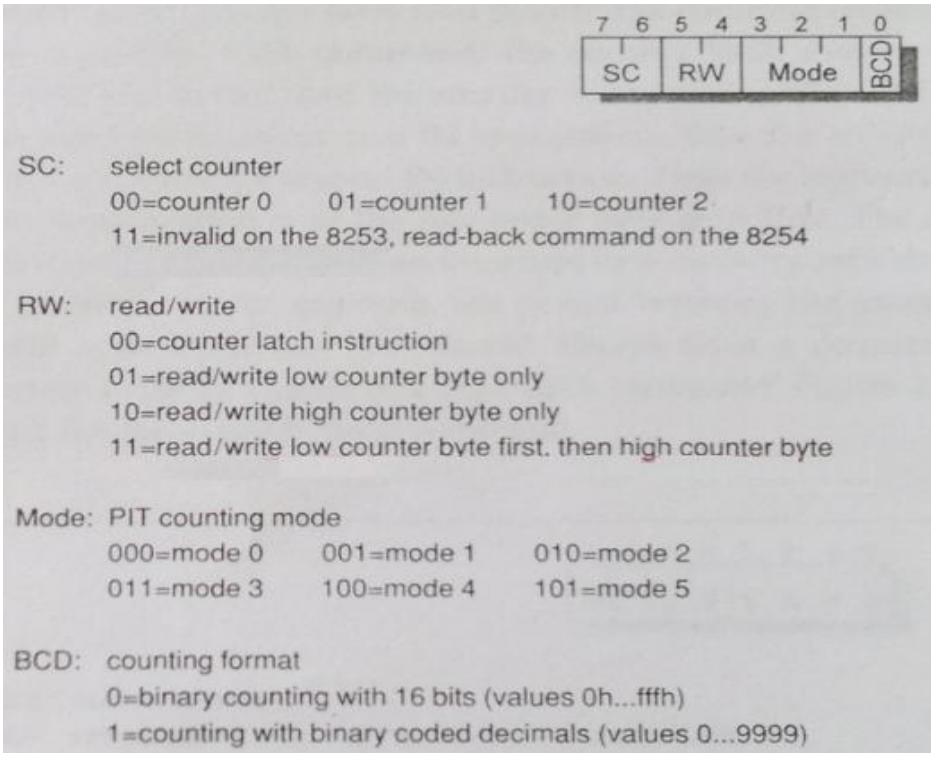


Şekil 1: 8254 PIC'in iç yapısı

PIC her biri ayrı programlanabilen 16-bit genişliğinde üç adet (Counteer 0-2) sayıcılarına sahiptir. Her sayıcı zaman tabanı (time base) olarak hizmet veren kendi saat işaretine (CLK0-CLK2) sahiptir. Saymayı tetiklemek veya başlatmak için GATE0-GATE2 kapı işaretlerinden yararlanır. Sayma moduna bağlı olarak alçak-yüksek veya yüksek-alçak geçişlerinde sayıcı aktif yapılır ve OUT0-OUT2 çıkışlarında modun gerektirdiği biçimde bir işaret üretilir.

8254 PIC'in PROGRAMLANMASI

Programlamak için ilk olarak kontrol kelimesi kontrol kaydediciye yazılır, ve daha sonra bir veya iki veri baytı amaçlanan sayıcının portuna yazılır. Kontrol kaydedici bir kere yüklendikten sonra kontrol kaydediciye erişmeden sayıcılara başka değerler yazılabilir. 8253 PIC'in kontrol kaydedicisi sadece yazılan, 8254'ün kaydedicisi ise hem yazılan hem de okunan bir kaydedicidir.



Şekil 2: Kontrol kaydedicisi

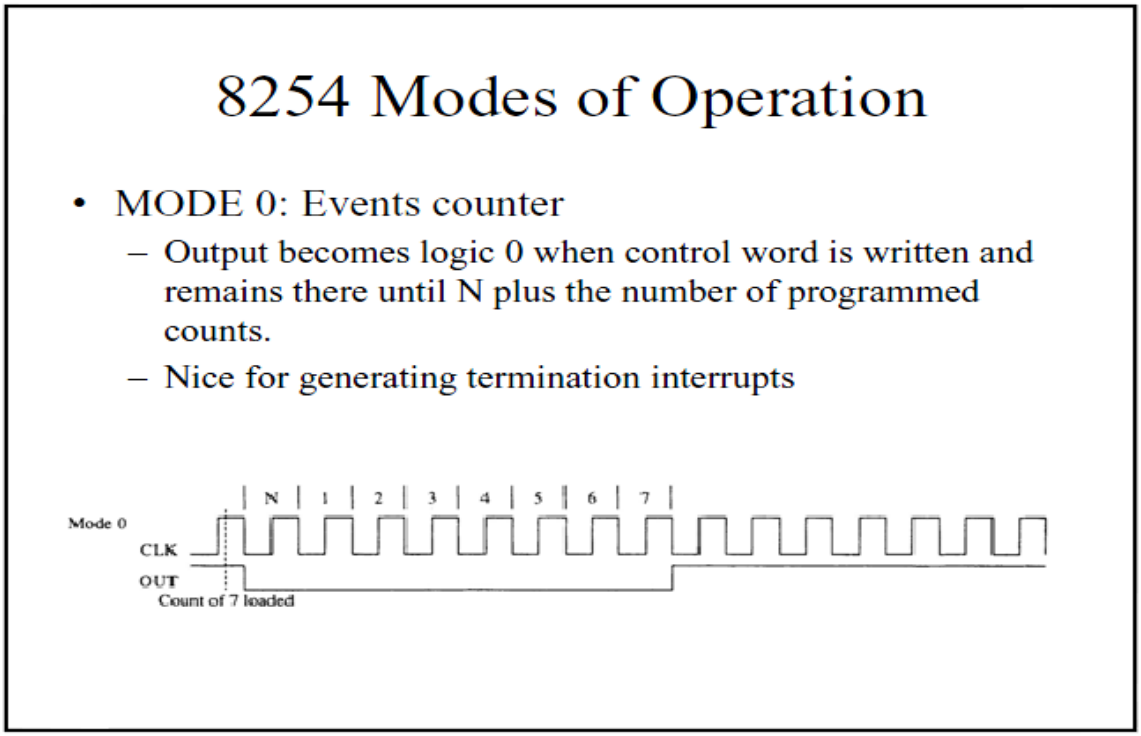
SAYMA MODLARI

8254 PIC altı farklı sayma moduna sahiptir ve hem ikili hem de onlu moda sayabilir. Çeşitli çalışma modları aşağıdaki şekillerde gösterilmiştir. Tüm modlarda PIT ilk sayma değerinden başlayarak aşağı değerlere kadar sayar. Yeni sayma değerleri herhangi bir anda sayıcılara yazılabilir. Sayıcı 0 değerine eriştiği zaman 0,1,4, 5 periyodik olmayan modlarda sayıcı saymayı durdurmaz, ama FFFFh (BCD=0) veya 9999 (BCD=1) ile devam eder.

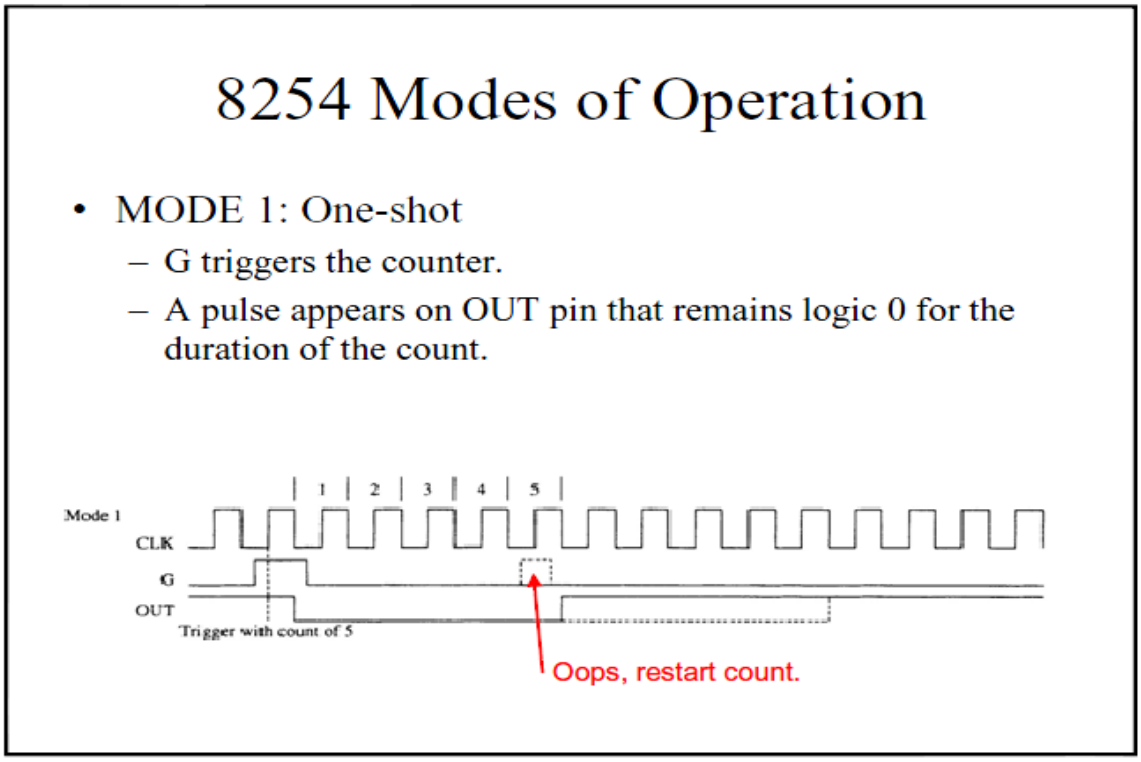
Mod 0: Nihai değerde kesme üretir.
Mod 1: Programlanabilen monoflop
Mod 2: Oran üretici
Mod 3: Kare-dalga üreteci

Mod 4: Yazılım tetiklemeli darbe

Mod 5: Donanım tetiklemeli darbe



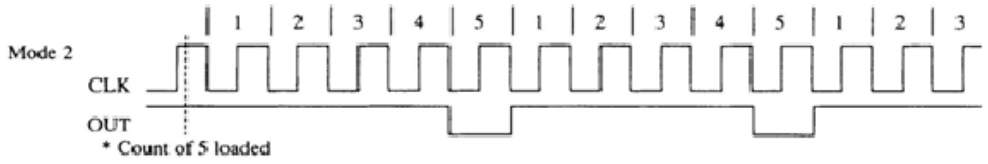
Şekil 3: Mod 0 zamanlama diyagramı



Şekil 4: Mod 1 zamanlama diyagramı

8254 Modes of Operation

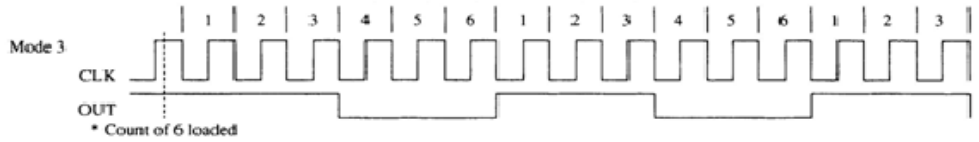
- MODE 2: Programmable clock generator
  - Produces a pulse, one clock width wide, that is spaced based on the count.
  - Continues until G goes low or a new mode is programmed in.



Şekil 5: Mod 2 zamanlama diyagramı

8254 Modes of Operation

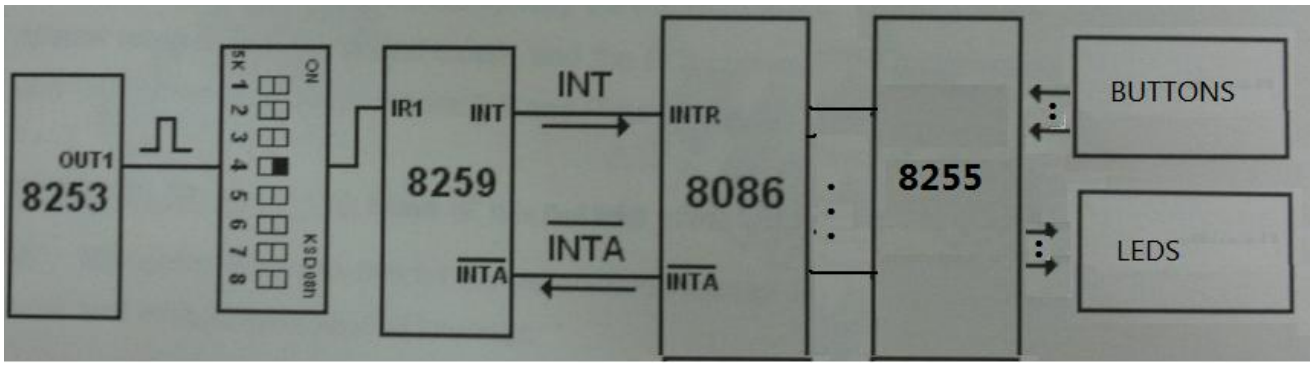
- MODE 3: Square wave generator
  - Generates a continuous square-wave at OUT as long as G is high.
  - If count is EVEN, OUT is hi for ½ the count, low for the other ½.
  - If count is ODD, the output is high for one extra clock period.



Şekil 6: Mod 3 zamanlama diyagramı

DENEYİN YAPILIŞI (Deney 1)

Amaç: Programın amacı 8253 PIT'ının 2. Zamanlayıcısını mod 3 te çalıştırarak frekansı değiştirilebilen bir kare dalga üreteci yapmaktır. Üretilen kare dalganın frekansını değiştirmek için 8255'in A portuna bağlı butonlar kullanılacaktır. Üretilen kare dalga sinya 8255'in B Portuna bağlı 8 adet led üzerinden dışarıya yansıtılacaktır.



Deney 1 devre şeması

Kaynak Program:

```
;================================karedalga.ASM=====================================
==

COMM1       EQU   0FFC8H          ;Define 8259 command address for
                                  ; ICW1, OCW2, OCW3
COMM2       EQU   0FFCAH          ; Define 8259 command address for
                                        ;ICW2, ICW3,ICW3, OCW1
COUNT1            EQU   0FFDAH          ; Define 8253 counter#1 port address
CNT_CSR           EQU   0FFDEH          ; Define 8253 control  word port address
CNT3              EQU   3FD6H           ;Define 8255 control word port address

APORT3            EQU   3FD0H           ;Define 8255 portA address
BPORT3            EQU   3FD2H           ;Define 8255 portB address
CPORT3            EQU   3FD4H           ;Define 8255 portC address



CODE        SEGMENT
            ASSUME  CS:CODE,DS:CODE

            ORG   0
```

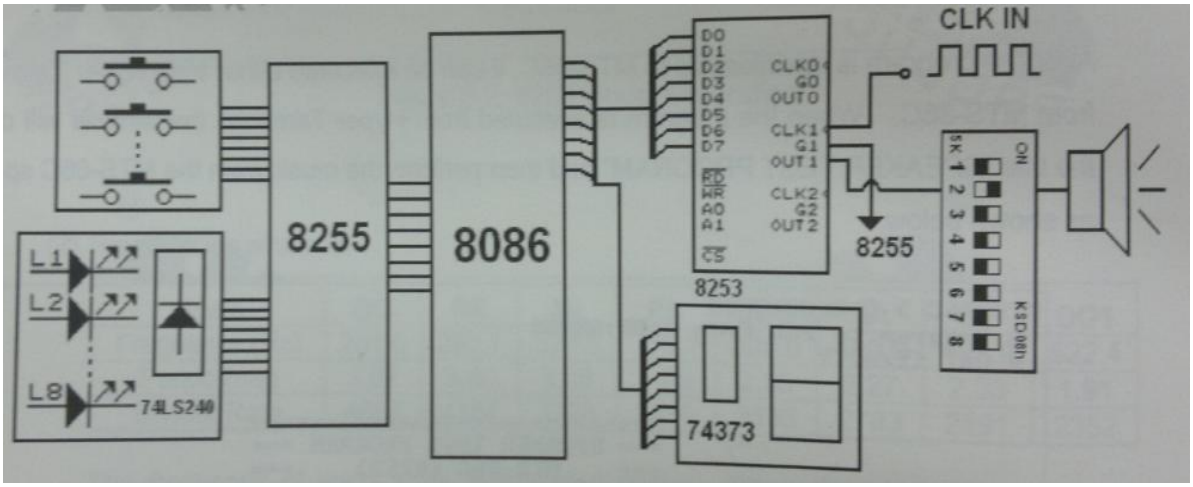
```
START:      CLI                         ;Disable interrupt

            MOV   DI,0                  ; Register for BPORT3 data
            MOV   SI,0                  ; Register for loop inside the subroutine

            MOV   AX,CS                 ;CS=DS
            MOV   DS,AX                 ;Code segment=Data Segment
            MOV   SP,0F000H             ;Setup stack pointer

;<Setup IR0  interrupt vector>

            MOV   AX,0                  ;İnitialize ES to '0'
            MOV   ES,AX                 ;Extra Segment =0

            MOV   BX,41H*4              ;Setup vector address of IR1 to BX=41H*4
            MOV   ES:WORD PTR[BX],OFFSET INTR1
                                        ;IP of ISR write to vector
            MOV   ES:2[BX],CS           ;CSP of ISR write to vector

;<İnitialize 8259 Command Words>
;<ICW1>
            MOV   DX,COMM1              ;Enable command 1 port address
            MOV   AL,00010011B          ; Setup ICW1
            OUT   DX,AL
;<ICW2>
            MOV   DX,COMM2              ;Enable command 2 port address
            MOV   AL,40H                ; Setup ICW3
            OUT   DX,AL

            MOV   AL,00000101B          ; Setup ICW4
            OUT   DX,AL

;<8259 operation command words>
;<OCW1>
            MOV   AL,11111101B          ;OCW1(Unmask IR1)
            OUT   DX,AL

;<Setup 8255 control word register>

            MOV   AL,90H                ;AL=90, portA=input, portB=portC=output
            MOV   DX,CNT3               ;Enable 8255 control port
            OUT   DX,AL
```

```
                MOV   AL,0FFH          ;Setup output to logic High
                MOV   DX,CPORT3        ;Enable 8255 portC
                OUT   DX,AL            ;Output FFh so that Gate1 of 8253 =High

;<setup 8253 control word register>

I8253:          MOV   DX,CNT_CSR       ;Enable 8253 control port
                MOV   AL,01110110B     ;setup 8253 control word register
                OUT   DX,AL
;<Output to 8253>
                MOV   DX,COUNT1        ;Enable 8253 count#1
                MOV   AX,0FFFFH        ;Divisor set to FFFFh
                OUT   DX,AL            ;Low Byte of divisor transfer to 8253
                MOV   AL,AH            ;High byte of divisor transfer to AL
                OUT   DX,AL            ; High Byte of divisor transfer to 8253

                STI                    ;Enable interrupt
;<Main code>

TEKRAR:         MOV   DX,APORT3        ; Enable 8255 portC
                IN    AL,DX            ;Read button data
                MOV   AH,AL            ;Transfer AL to AH
                MOV   AL,0FFH          ;Write FFh to AL

                MOV   DX,COUNT1         ;Enable 8253 count#1
                OUT   DX,AL            ;Low Byte of divisor transfer to 8253
                MOV   AL,AH            ;High byte of divisor transfer to AL
                OUT   DX,AL            ; High Byte of divisor transfer to 8253

                JMP   TEKRAR

;<IR1 Interrupt Service Routine>

INTR1:          PUSH  AX               ;Store Ax
                PUSH  DX               ;Store DX

                INC   SI               ;Increment SI to count timer interrupt calls
                CMP   SI,0010H         ; SI==10H?
                JNZ   INTR1_BITIR      ;No, then jump to INTR1_BITIR

                MOV   SI,0             ;Yes, clear SI (for recounting process)
                NOT   DI               ; Invert DI (which is the data to be transfered to BPORT3 )
                MOV   AX,DI            ;Transfer DI to AX
                MOV   DX,BPORT3        ;Enable BPORT3
```

```
                OUT     DX,AL           ;Write AL to BPORT3
INTR1_BITIR:
                MOV     DX,COMM1        ;Enable 8259 data port address (setup OCW2)
                MOV     AL,20H          ;EOI(End of Interrupt) Command write to 8259
                OUT     DX,AL           ;Output command

                POP     DX              ;Restore DX
                POP     AX              ;Restore AX

                IRET                    ;Return to main routine

CODE            ENDS
        END     START
```

DENEYİN YAPILIŞI (Deney 2)

Amaç: Program çalıştırıldığında, 8 TACT switch'ten girilen bilgi LED'lerde ve yedi parçalı display'de gösterilecek ve hoparlör aracılığıyla ilgili müzik skalasına dönüştürülecektir. Uygulamaya ilişkin devre şeması aşağıdaki gibidir.



Deney 2 devre şeması

Kaynak Program:

```
;================================piyano.ASM=======================================
COUNT1      EQU     0FFDAH          ;Define 8253 counter#1 port address
CSR         EQU     0FFDEH          ; Define 8253 control word port address
CNT3        EQU     3FD6H           ; Define 8255 control word port address
APORT3      EQU     3FD0H           ; Define 8255 portA address
BPORT3      EQU     3FD2H           ;Define 8255  portB address
```

```
CPORT3      EQU   3FD4H          ;Define 8255 portC address
FND         EQU   3FF0H          ; Define FND port address

            ORG   0

CODE        SEGMENT
            ASSUME CS:CODE,DS:CODE

START:      MOV   SP,4000H       ;Setup of stack pointer
            MOV   AX,CS          ;CS=DS
            MOV   DS,AX          ; Code Segment=Data segment

            MOV   AX,0           ;Initialize ES to '0'
            MOV   ES,AX          ;Extra segment=0
            MOV   BX,2*4         ;Setup vector address of NMI to BX=02H*4
            MOV   ES:WORD PTR[BX],OFFSET NMI
            MOV   ES:2[BX],CS    ;IP of ISR write to vector

            MOV   DX,CNT3        ;CS of ISR write to vector
            MOV   AL,90H         ;A PORT=Input, BC PORT=Output
            OUT   DX,AL          ;Output 90H to 8255 control port

            MOV   DX,CPORT3      ;Enable 8255 portC
            MOV   AL,0FFH        ;Setup output to logic High
            OUT   DX,AL          ;Output FFh so that Gate1 of 8253=High

;<Setup 8253 control word register>

I8253:      MOV   DX,CSR         ;Enable 8253 control port
            MOV   AL,01110110B   ;Setup 8253 control Word register
            OUT   DX,AL          ;Output data to 8253 control port
            MOV   DX,COUNT1      ;Enable 8253 count#1

;<Send Input data to LED>

PLAY:       PUSH  DX             ;DX=COUNT1, store to stack
            MOV   DX,APORT3      ;Enable 8255 PortA (8 bit TACT Switch)
            IN    AL,DX          ;Input data from TACT switch
            NOT   AL             ;Reverse the signal from Low to High
            MOV   DX,BPORT3      ;Enable 8255 PortB (8 bit LED)
            OUT   DX,AL          ;Output data to LED
            POP   DX             ;DX=COUNT1, load from stack

;<Pressed key changed?>
```

```
        CMP   AL,BL          ;When BL=unknown, ZF=0
                             ;When BL=AL, ZF=1
        JZ    PLAY           ;If ZF=0 ➔ key change, go next instruction
                             ;If ZF=1 ➔ no key change, jump to PLAY wait
                             ;for next key to be pressed.
        MOV   BL,AL          ;Copy AL to BL

;< Key pressed?>

        CMP   AL,0           ;When any key pressed ➔ ZF=0
                             ;When no key pressed ➔ ZF=1
        JZ    PLAY           ;If ZF=0 ➔ key pressed, go to next instruction
                             ;  ➔ play music and show in FND
                             ;If ZF=1 ➔ no key pressed, jump to PLAY wait
                             ; for next key to be pressed

;< Short period of soundless>

        PUSH  AX             ;Store input data from TACT switch to stack
        MOV   AX,10          ;Data (Divisor) for soundless
        OUT   DX,AL          ;The divisor=10, the frequency is too high
        MOV   AL,AH          ;to be heard
        OUT   DX,AL
        MOV   CX,1500        ;Time delay
        LOOP  $
        POP   AX             ;Load input data back to AX from stack

;< Determine which key is pressed>

        PUSH  DX             ;DX=COUNT1, store to stack
        MOV   DX,FND         ;Enable FND port address

        TEST  AL,80H         ;"7" key pressed? SW10.7=!7FH=80H
        JNZ   DO             ; If pressed, jump to DO.
        TEST  AL,40H         ;"6" key pressed? SW10.6=!BFH=40H
        JNZ   RE             ; If pressed, jump to RE.
        TEST  AL,20H         ;"5" key pressed? SW10.5=!DFH=20H
        JNZ   MI             ; If pressed, jump to MI.
        TEST  AL,10H         ;"4" key pressed? SW10.4=!EFH=10H
        JNZ   FA             ; If pressed, jump to FA.
        TEST  AL,8           ;"3" key pressed? SW10.3=!F7H=08H
        JNZ   SOL            ; If pressed, jump to SOL.
        TEST  AL,4           ;"2" key pressed? SW10.2=!FBH=04H
```

```
        JNZ     RA              ; If pressed, jump to RA.
        TEST    AL,2            ;"1" key pressed? SW10.1=!FDH=02H
        JNZ     SY              ; If pressed, jump to SY.
        TEST    AL,1            ;"0" key pressed? SW10.0=!FEH=01H
        JNZ     DO1             ; If pressed, jump to DO1.

DO:     MOV     AL,11011000B;   ;Number "7" for FND
        OUT     DX,AL           ;Output to FND
        MOV     AX,4697         ;Divisor for "DO"
        JMP     SET8253         ;Jump to SET8253

RE:     MOV     AL,10000010B;   ;Number "6" for FND
        OUT     DX,AL           ;Output to FND
        MOV     AX,4184         ;Divisor for "RE"
        JMP     SET8253         ;Jump to SET8253

MI:     MOV     AL,10010010B;   ;Number "5" for FND
        OUT     DX,AL           ;Output to FND
        MOV     AX,3728         ;Divisor for "MI"
        JMP     SET8253         ;Jump to SET8253

FA:     MOV     AL,10011001B;4  ;Number "4" for FND
        OUT     DX,AL           ;Output to FND
        MOV     AX,3519         ;Divisor for "FA"
        JMP     SET8253         ;Jump to SET8253

SOL:    MOV     AL,10110000B;3
        OUT     DX,AL           ;Output to FND
        MOV     AX,3135         ;Divisor for "SOL"
        JMP     SET8253         ;Jump to SET8253

RA:     MOV     AL,10100100B    ;Number "2" for FND
        OUT     DX,AL           ;Output to FND
        MOV     AX,2793         ;Divisor for "RA"
        JMP     SET8253         ;Jump to SET8253

SY:     MOV     AL,11111001B    ;Number "1" for FND
        OUT     DX,AL           ;Output to FND
        MOV     AX,2491         ;Divisor for "RA"
        JMP     SET8253         ;Jump to SET8253

DO1:    MOV     AL,11000000B    ;Number "0" for FND
        OUT     DX,AL           ;Output to FND
        MOV     AX,2352         ;Divisor for High "DO"
```

```
;<Play Sound>
SET8253:    POP   DX          ;DX=COUNT1, load from stack
            OUT   DX,AL       ;Low Byte of divisor transfer to 8253
            MOV   AL,AH       ;High Byte of divisor transfer to AL
            OUT   DX,AL       ;High Byte of divisor transfer to 8253

            MOV   CX,1500     ;Time delay
            LOOP  $
            JMP   PLAY
;<Interrupt Service Routine for NMI>

NMI:        PUSH  DX          ;DX store to stack
            PUSH  AX          ;AX store to stack
            MOV   DX,COUNT1   ;Enable 8253 Count1
            MOV   AX,10       ;Data (Divisor) for soundless
            OUT   DX,AL       ;The divisor= 10, frequency too high to be heard
            MOV   AL,AH
            OUT   DX,AL
            POP   AX          ;AX return from stack
            POP   DX          ;DX return from stack
            IRET

CODE        ENDS
            END   START
```