Dec	Hex	Bin
13	D	00001101

ORG ; THIRTEEN

8253/54 Timer

The x86 PC

assembly language, design, and interfacing

fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY

The x86 PC assembly language, design, and interfacing

fifth edition

Prentice Hall

OBJECTIVES

this chapter enables the student to:

- Describe the function of each pin of the 8253/54 PIT. (programmable interval timer)
- Program the three counters of the 8253/54 by use of the chip ’s control word.
- Diagram how the 8253/54 timer is connected in the IBM PC.
- Write programs to play music notes on the x86 PC speaker.

13.1: 8253/54 TIMER
8253 compared to 8254

- 8254 replaced the 8253 starting with the PC/AT.
  - The two have exactly the same pinout.
    - 8254 is a superset of the 8253.

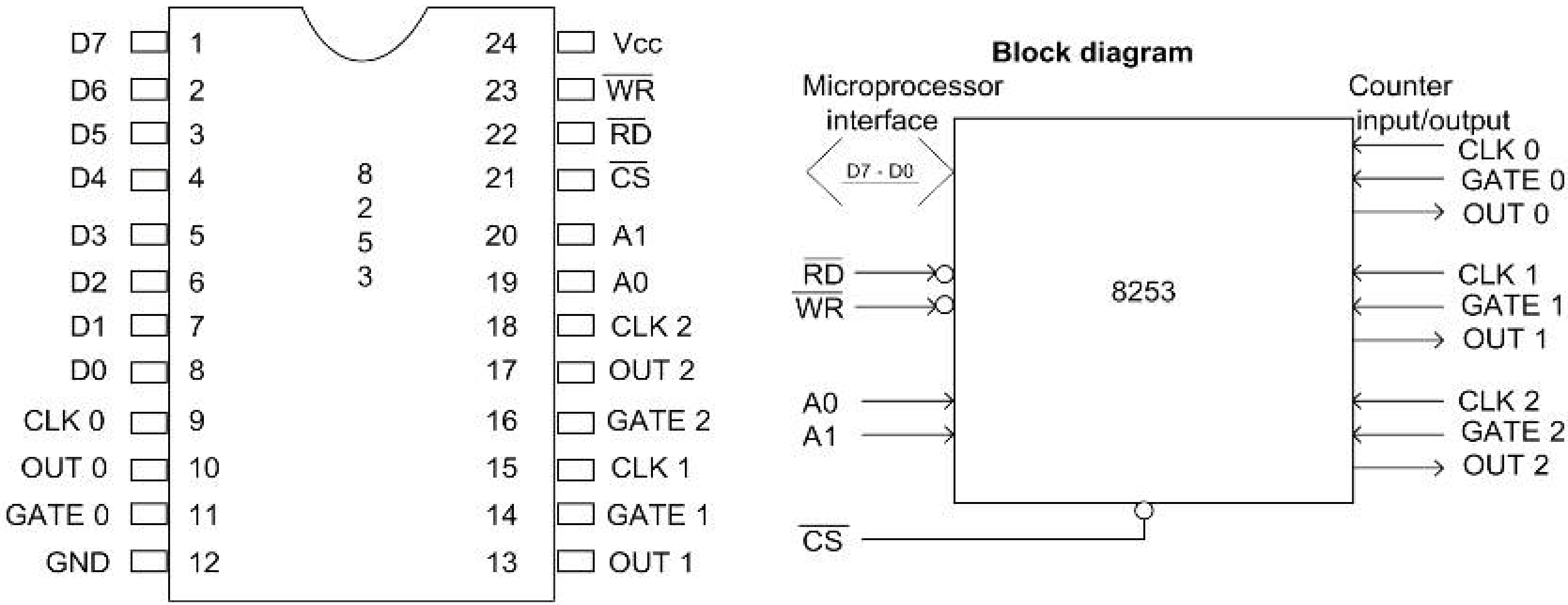


Figure 13-1 8253 Pin and Function Diagram

13.1: 8253/54 TIMER pins

- **A0**, **A1**, **CS** - three independent counters which are programmed separately to divide input frequency by a number from 1 to 65,536.
  - Each counter is assigned an individual port address.
  - The control register common to all three counters has its own port address.

A total of 4 ports are needed for a single 8253/54 timer.

Ports are addressed by **A0**, **A1**, and **CS**.

Table 13-1: Addressing 8253/54

$\overline{CS}$	A1	A0	Port
0	0	0	Counter 0
0	0	1	Counter 1
0	1	0	Counter 2
0	1	1	Counter register
1	x	x	8253/54 is not selected

13.1: 8253/54 TIMER pins CLK, GATE, OUT

Each counter has pins **CLK** (clock), **GATE** & **OUT**

Example 13-1

Pin CS of a given 8253/54 is activated by binary address A7–A2 = 100101.
(a) Find the port addresses assigned to this 8253/54.
(b) Find the configuration for this 8253/54 if the control register is programmed as follows.

```
MOV    AL,00110110
OUT    97H,AL
```

Solution:

(a) From Table 13-1, we have the following:

CS	A1A0	Port	Port address (hex)
1001 01	0 0	Counter 0	94
1001 01	0 1	Counter 1	95
1001 01	1 0	Counter 2	96
1001 01	1 1	Control register	97

(b) Breaking down the control word 00110110 and comparing it with Table 13-1 indicates counter 0 since the SC bits are 00. The RL bits of 11 indicate that the low-byte read/write is followed by the high byte. The mode selection is mode 3 (square wave), and finally binary counting is selected since the D0 bit is 0.

13.1: 8253/54 TIMER pins

- **CLK** - input clock frequency, between 0 & 2 MHz for 8253 and high as 10 MHz for 8254-2.
- **OUT -** after being divided, the shape of the output frequency from this pin can be programmed.
  - Square wave, one shot, etc.
  - No sine wave or saw tooth shapes.
- **GATE** - enables/disables the counter.
  - *HIGH* (5 V) on **GATE** enables; *LOW* (0 V) disables.
    - In some modes, a 0 to 1 pulse must be applied to enable.

13.1: 8253/54 TIMER pins

- **D0–D7** - a bidirectional bus connected to **D0–D7** of the system data bus.
  - Allows CPU access to registers inside 8253/54 for both read and write operations.
  - **RD** & **WR** (both *active-low*) are connected to **IOR** & **IOW** of the system bus.

13.1: 8253/54 TIMER
initialization of 8253/54

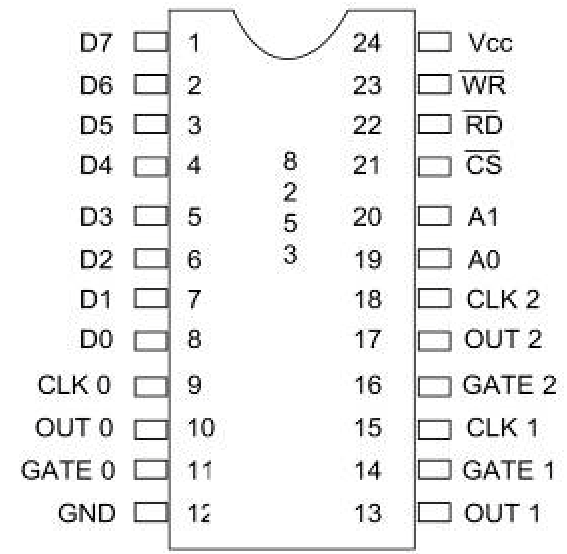
8253/54 must be initialized before it is used.

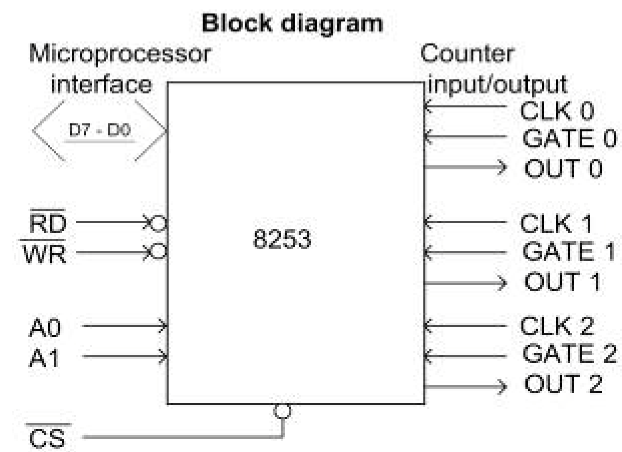
Each of the three counters must be programmed separately.

The control byte must first be written into the control register.

The number the input clock should be divided by must be written into that counter

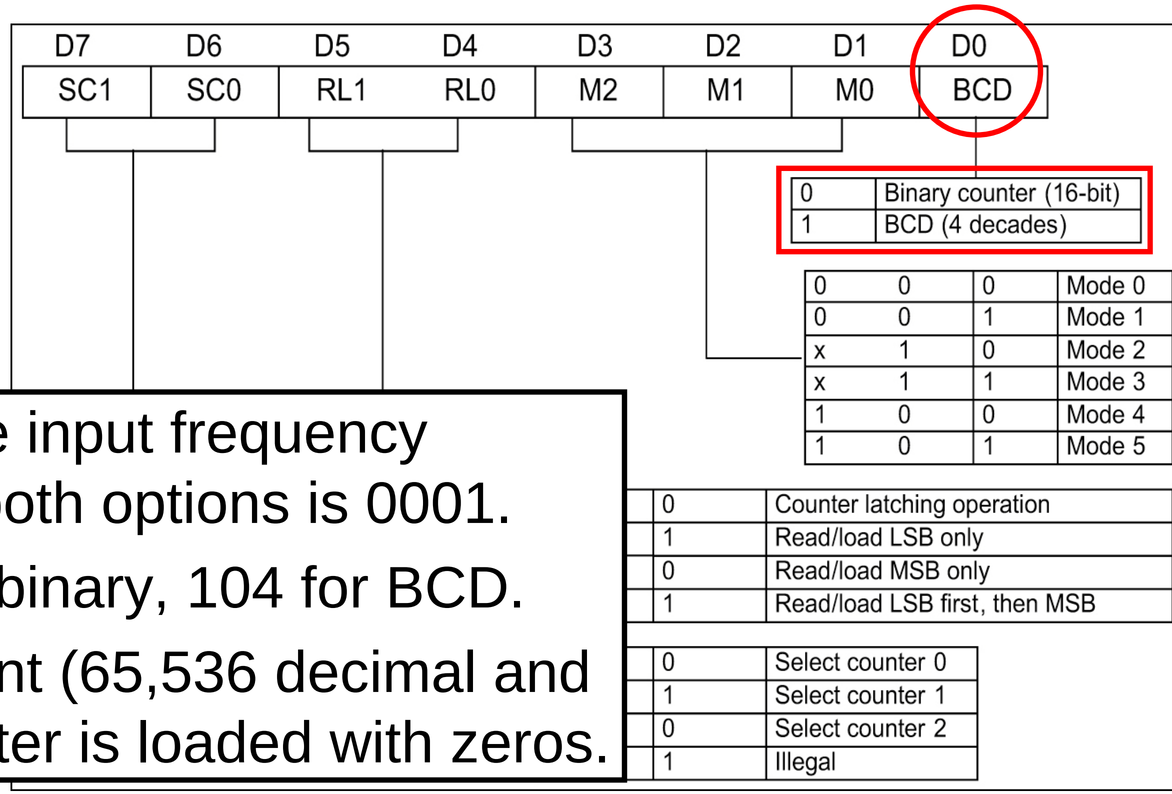
This can be as high as FFFF, which is (16-bit data), so the divisor must be sent one byte at a time.





The one-byte control word of the 8253/54, sent to the control register, has the following bits:

D0 chooses between a binary number divisor of 0000 to FFFFH or a BCD divisor of 0000 to 9999H.



The lowest number the input frequency can be divided by for both options is 0001.

The highest is 216 for binary, 104 for BCD.

To get the highest count (65,536 decimal and 10000 BCD), the counter is loaded with zeros.

Figure 13-2 8253/54 Control Word Format

13.1: 8253/54 TIMER control word

The one-byte control word of the 8253/54, sent to the control register, has the following bits:

D1, **D2**, and **D3** are for mode selection.

Six possible modes determine output signal shape.

Mode 0 Interrupt on terminal count.
Mode 1 Programmable one shot.
Mode 2 Rate generator.
Mode 3 Square wave rate generator.
Mode 4 Software triggered strobe.
Mode 5 Hardware triggered strobe.

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

0	Binary counter (16-bit)
1	BCD (4 decades)

0	0	0	Mode 0
0	0	1	Mode 1
x	1	0	Mode 2
x	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

0	0	Counter latching operation
0	1	Read/load LSB only
1	0	Read/load MSB only
1	1	Read/load LSB first, then MSB

0	0	Select counter 0
0	1	Select counter 1
1	0	Select counter 2
1	1	Illegal

Figure 13-2 8253/54 Control Word Format

13.1: 8253/54 TIMER control word

The one-byte control word of the 8253/54, sent to the control register, has the following bits:

D4 & **D5** are for **RL0** & **RL1** - to indicate size of the input frequency divisor, with 3 options:

- Read/write the most significant byte (MSB) only.
- Read/write the least significant byte (LSB) only
- Read/write LSB first, followed immediately by MSB

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

0	Binary counter (16-bit)
1	BCD (4 decades)

0	0	0	Mode 0
0	0	1	Mode 1
x	1	0	Mode 2
x	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

0	0	Counter latching operation
0	1	Read/load LSB only
1	0	Read/load MSB only
1	1	Read/load LSB first, then MSB

0	0	Select counter 0
0	1	Select counter 1
1	0	Select counter 2
1	1	Illegal

Figure 13-2 8253/54 Control Word Format

13.1: 8253/54 TIMER
control word

The one-byte control word of the 8253/54, sent to the control register, has the following bits:

D6 & **D7** are used to select which of the three counters is to be initialized by the control byte, **0**, **1**, **2**.

To program a given counter to divide the **CLK** input frequency, send the divisor to that specific counter's register.

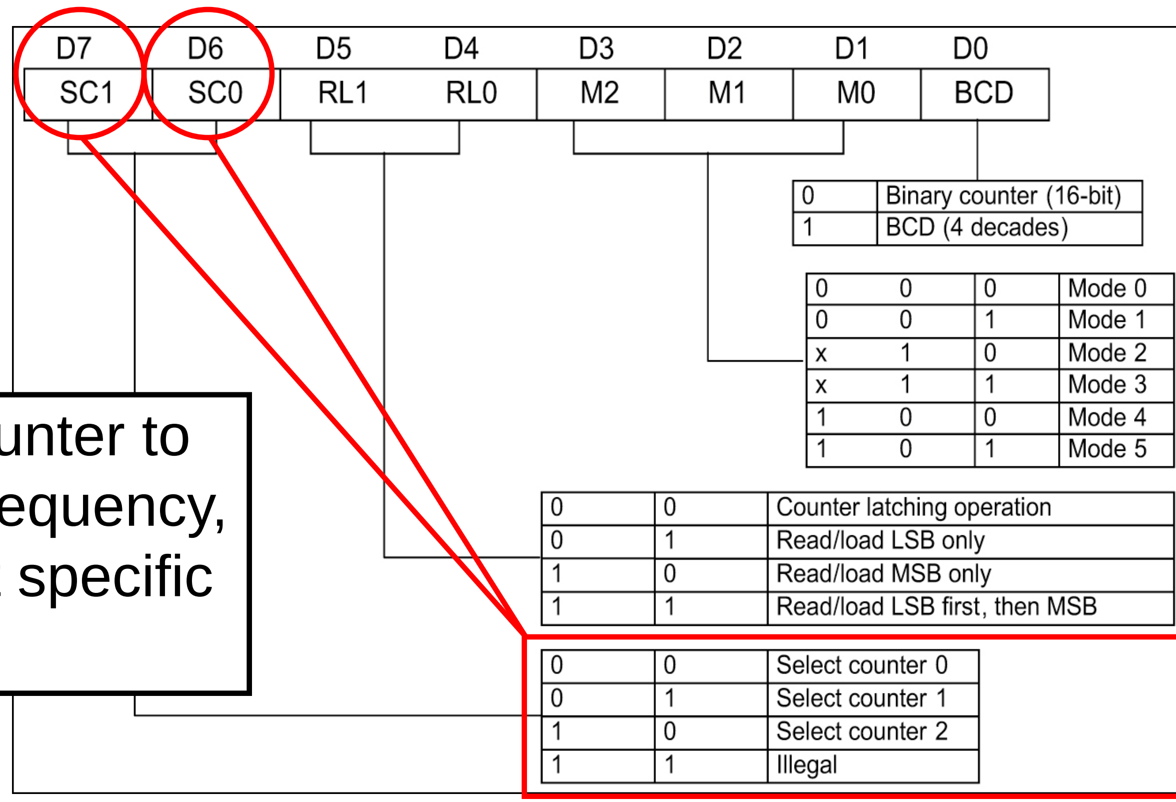


Figure 13-2 8253/54 Control Word Format

13.1: 8253/54 TIMER control word options

- In BCD mode, if the counter is programmed for 9999, the input frequency is divided by that number.
  - To divide the frequency by 10,000, send in 0 for *both* high *and* low bytes.

Example 13-2

Using the port addresses in Example 13-1, show the programming of counter 1 to divide CLK1 by 10,000, producing the mode 3 square wave. Use the BCD option in the control byte.

Solution:

```
MOV   AL,77H     ;counter 1, mode 3, BCD
OUT   97H,AL     ;send it to control register
SUB   AL,AL      ;AL = 0 load the divisor for 10,000
OUT   95H,AL     ;send the low byte
OUT   95H,AL     ;and then the high byte to counter 1
```

See also example 13-2 on page 353 of your textbook.

13.2: x86 8253/54 CONNECTION/PROGRAMMING

The first PC used a 74LS138 to decode addresses for CS of the 8253.

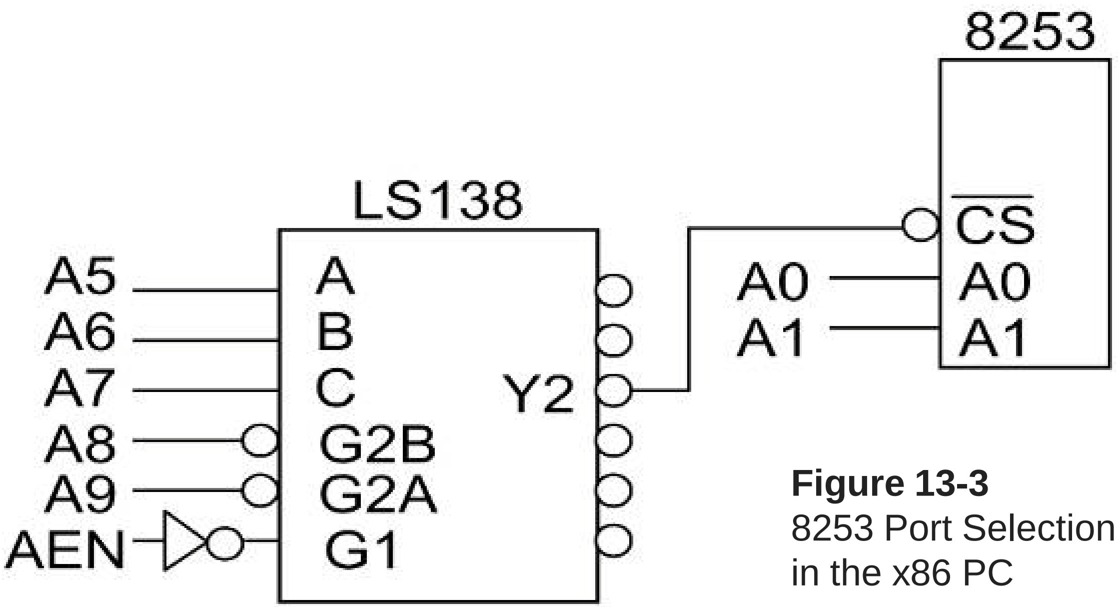


Figure 13-3
8253 Port Selection in the x86 PC

Port addresses are selected as indicated in Table 13-2.

(assume zeros for x's)

Table 13-2: 8253/54 Port Address Calculation in the x86 PC

Binary Address			Hex Address	Function
$\overline{AEN}$ A9 A8	A7 A6 A5 A4	A3 A2 A1 A0		
1 0 0	0 1 0 x	x x 0 0	40	Counter 0
1 0 0	0 1 0 x	x x 0 1	41	Counter 1
1 0 0	0 1 0 x	x x 1 0	42	Counter 2
1 0 0	0 1 0 x	x x 1 1	43	Counter register

13.2: x86 8253/54 CONNECTION/PROGRAMMING clocks

- The **three clocks** of the 8253 are all connected to a constant frequency of 1.1931817 MHz.
  - From **PCLK** of the 8284 chip after it has been divided by 2 with the use of **D flip-flop** 72LS175.

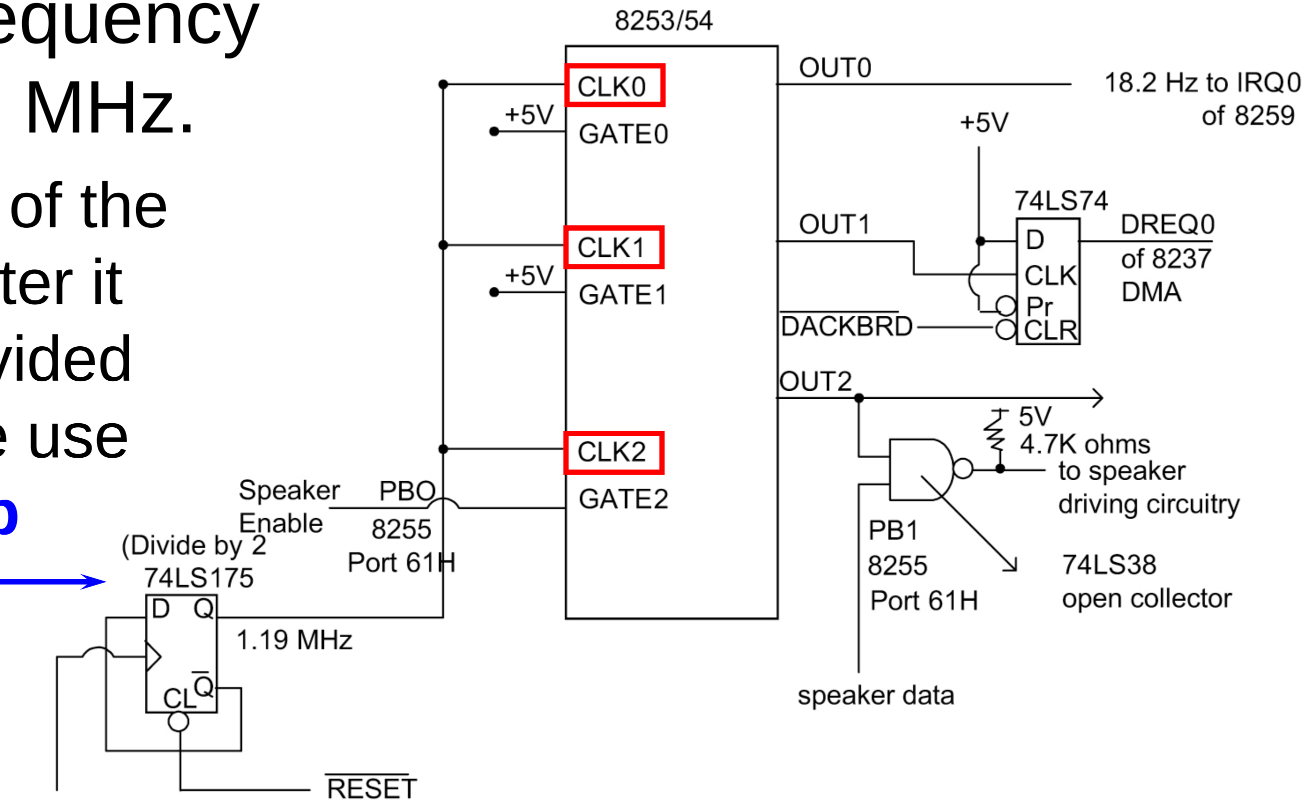


Figure 13-4 8253/54 Chip Connections in the x86 PC

13.2: x86 8253/54 CONNECTION/PROGRAMMING clocks

- **CLK0** of counter 0 is 1.193 MHz, and **GATE0** is connected to *high* permanently.
  - **OUT0** of counter 0 is connected to **IRQ0** (highest priority interrupt) of the 8259 interrupt controller.
- **IRQ0** is activated 18.2 times per second, making the **OUT0** frequency 18.2 Hz.
- The wave shape is a square wave. (8253, mode 3)
  - Triggers **IR0** on the positive edge of each pulse of the square wave.
    - So a high pulse is not mistaken for multiple interrupt.

13.2: x86 8253/54 CONNECTION/PROGRAMMING using counter 0 – the control word

- **D0 = 0** - binary (or hex) value of the counter divisor.
  - The timer decrements after every input pulse, to zero
    - Then the original value is loaded again.
  - To divide the input frequency by 65,536, the timer is programmed with 0s for both high *and* low bytes.

13.2: x86 8253/54 CONNECTION/PROGRAMMING using counter 0

- **D3**, **D2**, **D1** = 011, mode 3, for the square wave output of 18.2 Hz frequency.
- **D4**, **D5** = 11 for reading/writing the LSB first, followed by the MSB.
- **D7**, **D6** = 00 for counter.

D7	D6	D5	D4	D3	D2	D1	D0	
0	0	1	1	0	1	1	0	= **36H**

– Programming of counter 0.

```
MOV    AL,36H       ;control word
OUT    43H,AL       ;to control register of 8253
MOV    AL,00        ;00 LSB and MSB of the divisor
OUT    40H,AL       ;LSB to timer 0
OUT    40H,AL       ;MSB  to timer 0
```

13.2: x86 8253/54 CONNECTION/PROGRAMMING using counter 1

- In counter 1, **CLK1** is connected to 1.193 MHz and **GATE** is *high* permanently.
  - **OUT1** generates a periodic pulse required to refresh DRAM, at least every 15us for each cell.
- In the PC, refreshing DRAM is done by 8237 DMA.
  - 8253 counter 1 informs DMA periodically.
- **OUT1** will provide DMA a pulse of approximately 15 us duration or 66,278 Hz.

13.2: x86 8253/54 CONNECTION/PROGRAMMING using counter 1

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

The control byte can be figured out as follows:

- **D0** = 0 for binary option
- **D3**, **D2**, **D1** = 010 for mode 2 shape output.
  - **OUT1** stays high for 18 pulses & goes low for one pulse.
- **D5**, **D4** = 01 for the LSB only.(byte is less than FF)
  - CLK1 is divided by 18. (18 is the LSB; no need for the MSB)
- **D7**, **D6** = 01 for counter 1

```
D7 ... D0
0101 0100 = 54H for the control word
```

13.2: x86 8253/54 CONNECTION/PROGRAMMING using counter 1

- The programming of the 8253 counter 1 in the IBM BIOS is listed as follows, with slight modifications for the sake of clarity:

```
MOV    AL,54H          ; the control word
OUT    43H,AL          ; to control register
MOV    AL,18           ;18 decimal, the divisor
OUT    41H,AL          ; to counter 1
```

13.2: x86 8253/54 CONNECTION/PROGRAMMING using counter 2

- Output of counter 2 is connected the speaker and to PC5 of the 8255.
  - In early XTs, also to the cassette circuitry.
  - As counter 2 is used in the PC to play music, it is important to understand counter 2 programming.

13.2: x86 8253/54 CONNECTION/PROGRAMMING
use of timer 2 by the speaker

- In the PC..
  - **CLK2** is connected to a frequency of 1.19318 MHz.
  - **GATE2** is programmed by **PB0** of port **61H** (port **B**).
    - The PC uses counter 2 to generate the beep sound.
- BIOS uses timer 2 for the beep sound, but it can be changed to play any musical note.
  - The beep sound has a frequency of 896 Hz, of mode 3. (square wave)
  - Dividing input frequency of 1.19318 MHz by 896 Hz gives 1331 (0533 hex) for the value to be loaded to counter 2.
    - The process of turning on the speaker is the same for all x86 PCs regardless of microprocessor used.

13.2: x86 8253/54 CONNECTION/PROGRAMMING
turn on speaker via PB0 & PB1 of port 61H

- **GATE2** must be high to provide **CLK** to timer 2.
  - This function is performed by **PB0** of port **61H**.
  - **OUT2** of timer 2 is **AND**ed with **PB1** of port **61H**, then input to the driving circuitry of the speaker.

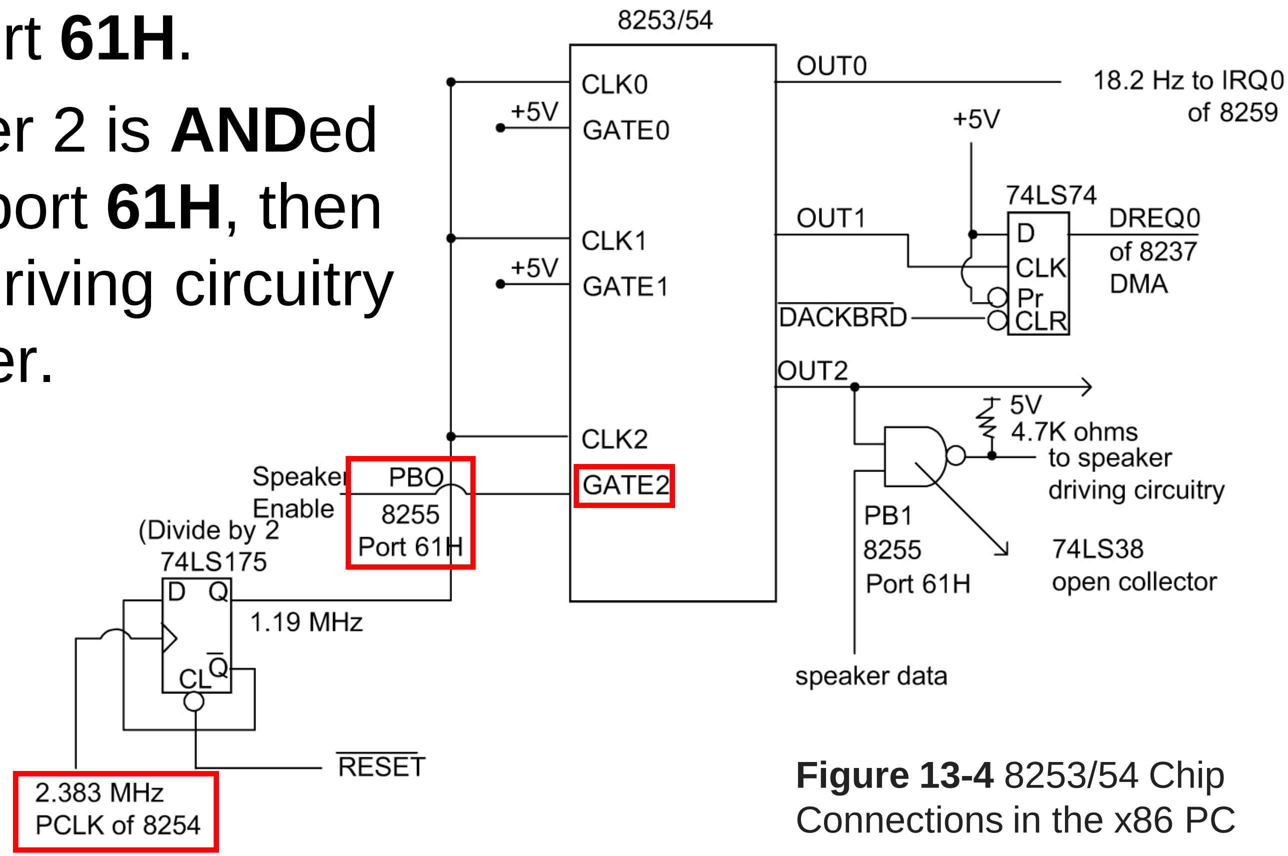


Figure 13-4 8253/54 Chip Connections in the x86 PC

13.2: x86 8253/54 CONNECTION/PROGRAMMING
turn on speaker via PB0 & PB1 of port 61H

- To allow **OUT2** to go to the speaker, **PB1** of port **61H** must be set to high.
  - The following is the code to turn the speaker on.
    - The same as the IBM BIOS's code to sound the BEEP.

```
IN     AL,61H          ;GET THE CURRENT SETTING OF PORT B
MOV    AH,AL           ;SAVE IT
OR     AL,00000011B    ;MAKE PB0=1 AND PB1=1
OUT    61H,AL          ;TURN THE SPEAKER ON.
{HOW LONG THE BEEP SHOULD SOUND GOES HERE}
MOV    AL,AH           ;GET THE ORIGINAL SETTING OF PORT B
OUT    61H,AL          ;TURN OFF THE SPEAKER
```

The amount of time a musical note plays is referred to as its *time delay*, produced with the help of the CPU in the x86 PC.

13.2: x86 8253/54 CONNECTION/PROGRAMMING
time delay for x86 PCs

- Using x86 instructions to generate time delays are is unreliable, as CPU speed varies among PCs.

```
;(CX) = COUNT OF 15.085
;15.085 MICROSECONDS
WAITF PROC   NEAR
      PUSH   AX
WAITF1:
      IN     AL,61H
      AND    AL,10H         ;CHECK PB4
      CMP    AL,AH     ;DID IT JUST CHANGE?
      JE     WAITF1    ;WAIT FOR CHANGE
      MOV    AH,AL     ;SAVE THE NEW PB4 STATUS
      LOOP   WAITF1    ;CONTINUE UNTIL CX BECOMES 0
      POP    AX
      RET
WAITF ENDP
```

To create a CPU-independent delay, x86 toggles **PB4** of port **61H**, every 15.085 microseconds.

13.2: x86 8253/54 CONNECTION/PROGRAMMING
time delay for x86 PCs

- A time delay of any duration can be created regardless of x86 CPU frequency.

Example 13-4

Using the BIOS WAITF routine, show how to create a 1.5-second time delay.

Solution:

Since the 1.5-second delay requires the counter to be set to 99,436 (1.5 s/15.085 μs = 99,436) and the maximum value of CX is 65,536, the following method is used to generate the 1.5-second delay.

```
        MOV   BL,03
BACK:   MOV   CX,33144    ;1/2-second delay
        CALL  WAITF
        DEC   BL
        JNZ   BACK
```

Monitor **PB4** of port **61H**, to obtain a fixed time delay.

- The input frequency to counter 2 is fixed at 1.1931817 MHz for all x86 PCs.
  - Programs for playing music found in this section can run on any of them without modification.

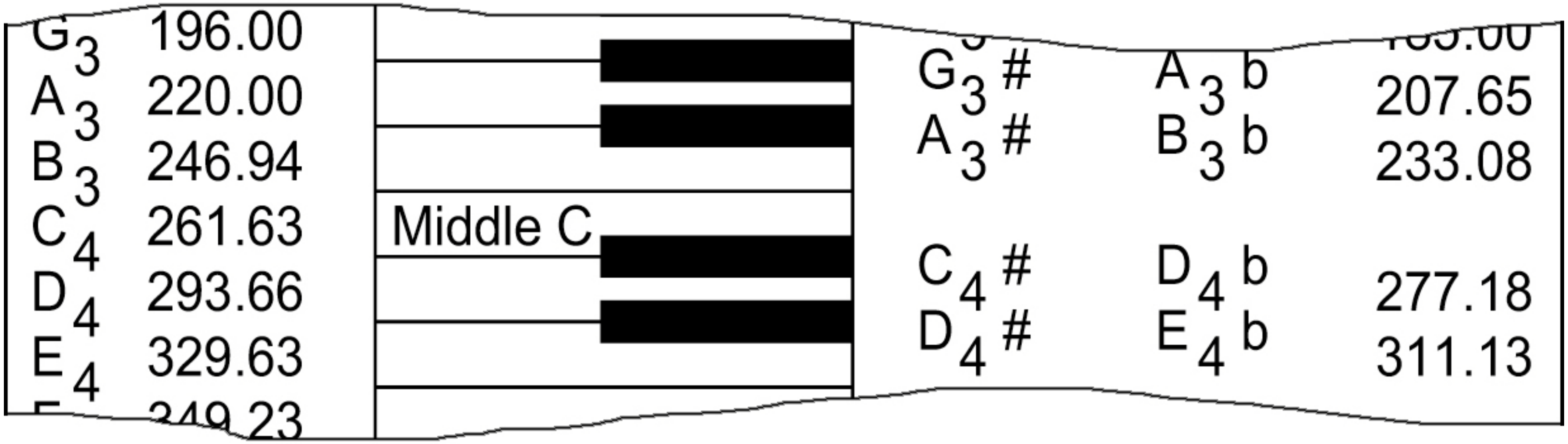
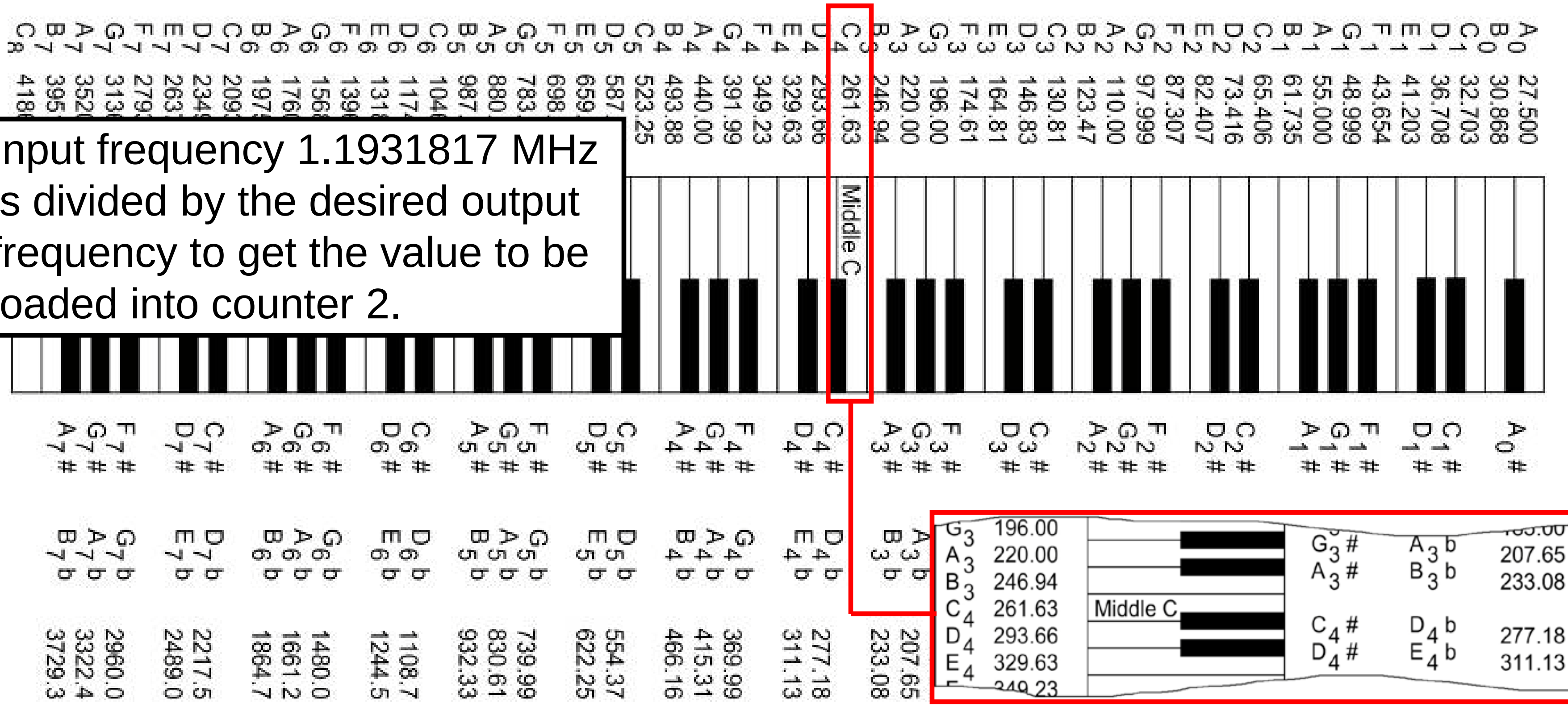


Fig. 12-5 Piano Note Frequencies

List of piano notes and frequencies.

Input frequency 1.1931817 MHz is divided by the desired output frequency to get the value to be loaded into counter 2.



Review this frequency chart in greater detail on page 360 of your textbook.

Fig. 12-5 Piano Note Frequencies

13.3: GENERATING MUSIC ON THE x86 PC

- Counter 2 is connected to the speaker and it can be programmed to output any frequency desired.

Example 13-5

Show the values to be loaded into counter 2 in order to have the output frequency for the notes (a) D3, (b) A3, and (c) A4.

Solution:

From Figure 13-5, notice that the frequency for note D3 is 147. The value that must be loaded into counter 2 is 1.1931 MHz divided by 147, which is 8116. Going through this procedure for each note gives the following:

		Value Loaded into Counter 2	
Note	Frequency	Decimal	Hex
D3	147 Hz	8116	1FB4
A3	220 Hz	5423	152F
A4	440 Hz	2711	0A97

Now that the values to be loaded into counter 2 are known, the program for getting the speaker to sound the notes for a certain duration is shown in Example 13-6.

Also review example 13-6 on page 361 of your textbook.

13.3: GENERATING MUSIC ON THE x86 PC

- For delays of 5ms and 250 ms in x86 PCs, the routines on page 362 can be used….

```
;
DELAY           PROC    NEAR
                MOV     CX,16578  ;16578 x 15.08 microsec = 250
ms
                PUSH    AX
WAIT:
                IN      AL,61H
                AND     AL,10H    ;check PB4
                CMP     AL,AH     ;did it just change?
                JE      WAIT      ;wait for change
                MOV     AH,AL     ;save the new PB4 status
                LOOP    WAIT      ;decrement CX and continue
                                  ;until CX becomes 0
                POP     AX
                RET
DELAY           ENDP
```

13.3: GENERATING MUSIC ON THE x86 PC

- For delays of 5ms and 250 ms in x86 PCs, the routines on page 362 can be used….

```
DELAY_OFF    PROC   NEAR
             MOV    CX,331      ;331 x 15.08 microsec = 5 ms
             PUSH   AX
WAIT:        IN     AL,61H
             AND    AL,10H      ;check PB4
             CMP    AL,AH       ;did it just change?
             JE     WAIT        ;wait for change
             MOV    AH,AL       ;save the new PB4 status
             LOOP   WAIT        ;continue until CX becomes 0
             POP    AX
             RET
DELAY OFF    ENDP
```

13.3: GENERATING MUSIC ON THE x86 PC
playing "Happy Birthday" on the PC

- This background should be sufficient to develop a program to play any song.
  - The tune for the song "Happy Birthday" is given.

Lyrics	Notes	Freq. (Hz)	Duration	Lyrics	Notes	Freq. (Hz)	Duration
hap	C4	262	1/2	hap	C4	262	1/2
py	C4	262	1/2	py	C4	262	1/2
birth	D4	294	1	birth	C5	523	1
day	C4	262	1	day	A4	440	1
to	F4	349	1	dear	F4	349	1
you	E4	330	2	so	E4	330	1
hap	C4	262	1/2	so	D4	294	3
py	C4	262	1/2	hap	B4b	466	1/2
birth	D4	294	1	py	B4b	466	1/2
day	C4	262	1	birth	A4	440	1
to	G4	392	1	day	F4	349	1
you	F4	349	2	to	G4	392	1
				you	F4	349	2

playing "Happy Birthday" on the PC

- The example program plays the first seven notes of the “Happy Birthday”.
  - Using arrays for notes and duration; tested using MASM.
  - In all examples, values loaded counter 2 were calculated by dividing 1.1931817 input to **CLK2** by the desired **OUT2** frequency.

See the entire program listing on pages 363-364 of your textbook.

```
;Tested by Danny Causey and Hanani Bonda
;This program plays the first 7 notes of "Happy Birthday"
        .MODEL SMALL
        .STACK 64
        .DATA
NOTES         DW    11CAH,11CAH,0FDAH,11CAH,0D5BH,0E1FH,11CAH
Duration      DB    2,2,4,4,4,8,2
        .CODE
START  PROC  FAR
        MOV   AX,@DATA
        MOV   DS,AX
        MOV   AL,0B6H  ;control byte: counter 2, LSB, MSB, binary
        OUT   43H,AL    ;send the control byte to control reg
        MOV   BX,7      ;set up counter
        LEA   SI,NOTES
        LEA   DI,Duration
AGAIN:MOV
```

13.3: GENERATING MUSIC ON THE x86 PC
generating music using C#

- x86 compilers restrict access to I/O ports using I/O instructions in Assembly language.
- Microsoft's .NET architecture provides an interface to achieve the same result

```
//C# of the first seven notes of the "Happy Birthday"
using System;
using System.Threading;
namespace Music
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.Beep(262, 500);   //hap for 500 ms
      Thread.Sleep(5);          //speaker off 5 ms
      Console.Beep(262, 500);   //py for 500 ms
      Thread.Sleep(5);          //speaker off 5 ms
      Console.Beep(294, 1000);  //birth for 1 s
      Thread.Sleep(5);          //speaker off 5 ms
      Console.Beep(262, 1000);  //day for 1 s
      Thread.Sleep(5);          //speaker off 5 ms
      Console.Beep(349, 1000);  //to for 1 s
      Thread.Sleep(5);          //speaker off 5 ms
      Console.Beep(330, 2000);  //you for 2 s
      Thread.Sleep(5);          //speaker off 5 ms
      Console.Beep(262, 500);   //hap for 500 ms
    }
  }
}
```

See the entire program listing on pages364 - 365 of your textbook.

13.3: GENERATING MUSIC ON THE x86 PC
generating music using C#

- Use these functions to generate the rest of the song for practice.
  - `Console.Beep(frequency, duration)` - will play the frequency for the duration in milliseconds.
    - Frequencies can range from 37 to 32767.
  - `Thread.Sleep(duration)`- will pause execution for the duration in milliseconds.

13.3: GENERATING MUSIC ON THE x86 PC
playing "Mary Had a Little Lamb"

- For further programming practice…
  - The tune for the song "Mary Had a Little Lamb".

Lyrics	Notes	Freq. (Hz)	Duration	Lyrics	Notes	Freq. (Hz)	Duration
hap	C4	262	1/2	hap	C4	262	1/2
py	C4	262	1/2	py	C4	262	1/2
birth	D4	294	1	birth	C5	523	1
day	C4	262	1	day	A4	440	1
to	F4	349	1	dear	F4	349	1
you	E4	330	2	so	E4	330	1
hap	C4	262	1/2	so	D4	294	3
py	C4	262	1/2	hap	B4b	466	1/2
birth	D4	294	1	py	B4b	466	1/2
day	C4	262	1	birth	A4	440	1
to	G4	392	1	day	F4	349	1
you	F4	349	2	to	G4	392	1
				you	F4	349	2

Also see this listing on page 366 of your textbook.

Dec	Hex	Bin
13	D	00001101

ENDS ; THIRTEEN