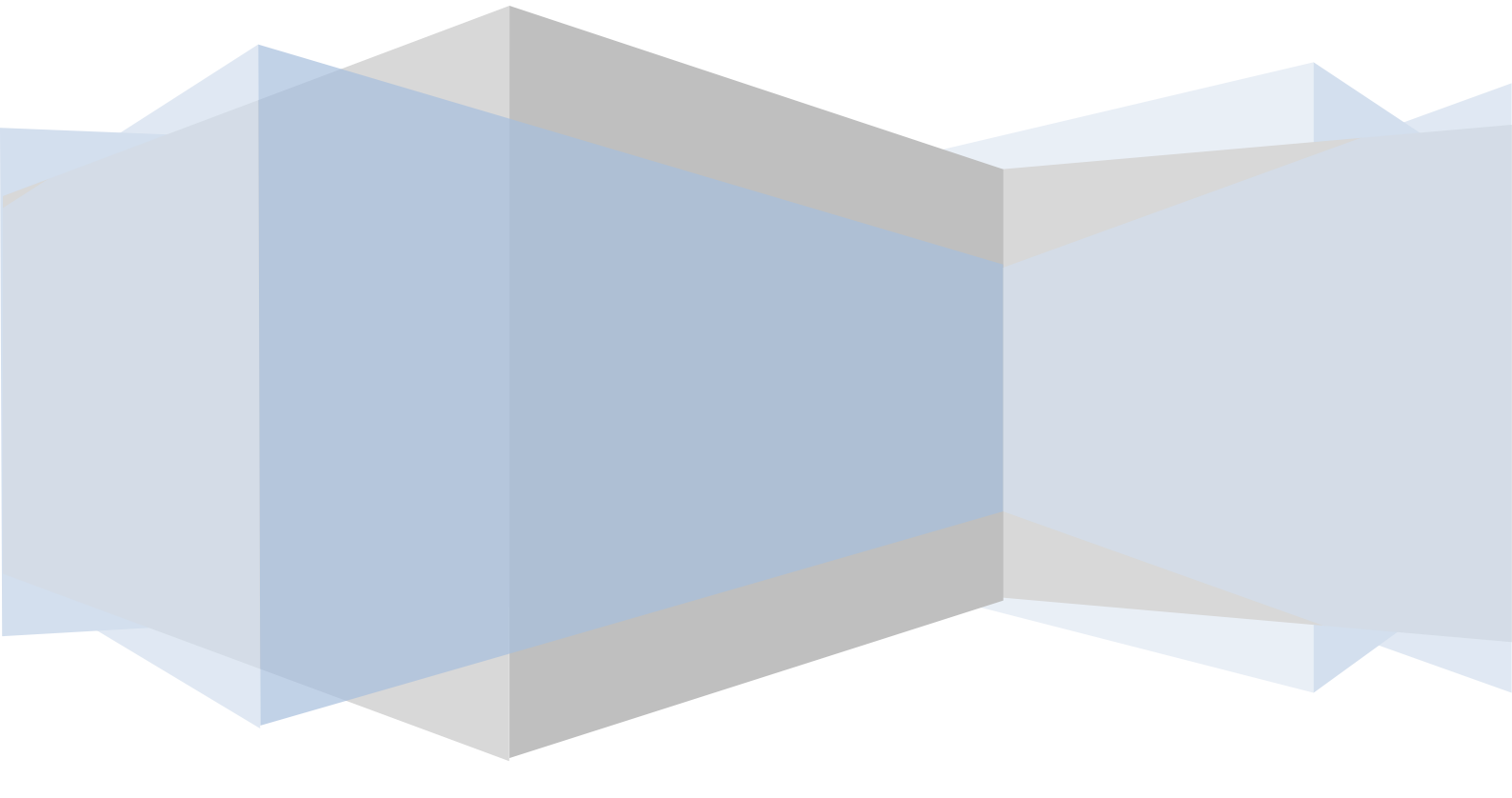
組合語言期末報告

電腦音樂的製作

資工二　　B95902126　　吳斌群



一、目的：開啓一個記錄著簡譜的檔案，並將該資料轉換爲音樂撥放出來。

二、原理：

(一)基本樂理：

音樂構成的兩大要素爲音符及節拍。以下分別討論：

1. 音符：

簡譜中以 1、2、3…表示 Do、Re、Mi…等七個中音。若在音符的上方或下方加入一點，則分別代表高八度音及低八度音。聲音的高低是由頻率來決定的。所以，只要想辦法讓音效晶片產生各種頻率的方波，就可以透過喇叭聽到不同的音調。下表列出各種音符與頻率的關係。

(Hz)	Do	Re	Mi	Fa	So	La	Si
第 0 八度音	65	73	82	87	98	110	123
第 1 八度音	131	147	165	175	196	220	247
第 2 八度音	262	294	330	349	392	440	494
第 3 八度音	523	587	659	698	784	880	988
第 4 八度音	1046	1175	1318	1397	1568	1760	1976
第 5 八度音	2093	2349	2637	2794	3136	3520	3951
第 6 八度音	4186	4699	5274	5587	6271	7040	7902

(1) 相鄰兩個八度音的頻率，正好相差二倍。

(2) 若遇到升半音(♯)和降半音(♭)的記號，頻率的決定方式如下：

5♯ → So 升半音 → 頻率=(784+880)/2=832Hz

5♭ → So 降半音 → 頻率=(784+698)/2=741Hz

(3) 休止符(0)在樂譜中代表不發音，我們可以將頻率提高到超過人耳所能接受的範圍，就聽不到聲音。所以休止符的頻率爲 65535Hz。

2. 節拍：

節拍是指每個音符發音的長短，下表列出樂譜中常見的拍數及名稱。

名稱	拍數	簡譜
全音符	4 拍	1 - - -
附點二分音符	3 拍	1 - -
二分音符	2 拍	1 -
四分音符	1 拍	1
八分音符	$\frac{1}{2}$ 拍	1
十六分音符	$\frac{1}{4}$ 拍	1
附點四分音符	$1\frac{1}{2}$ 拍	1 •
附點八分音符	$\frac{3}{4}$ 拍	1 •

(1) 在正常演奏速度下，每一拍持續的時間大約是 0.44 秒。

(2) 每個音符在演奏時，有 10/11 屬於發音部份，而有 1/11 則屬於不發音部份。在電腦的處理上可以分爲 Speaker ON(喇叭開啓)與 Speaker OFF(喇叭關閉)兩部份。

(3) 滑音是指兩相鄰的音符發音不間斷，類似圓滑而過的感覺。在電腦處理上可以設定 Speaker OFF

時間爲 0，讓整個音符全部發音，完畢後緊接著發下一個音。滑音的符號爲$\overset{\cap}{34}$。

(4) 歌譜左上方所出現的符號爲整首歌的演奏方式。

4/4 → 這首歌是以四分音符爲一拍，每小節有四拍。

3/4 → 這首歌是以四分音符爲一拍，每小節有三拍。

(二)樂譜表格之轉換：

爲了讓電腦演奏歌曲，必須將樂譜轉換爲電腦可以接受的表格資料。表格資料的格式定義爲每個音符以兩個位元組(Byte)表示。格式爲：

DB　　音符頻率，音符節拍，…

1. 音符頻率：參考音符與頻率的對照表，該位元組的高位數存放第幾八度音，低位數則表示 Do(1)、Re(2)、Mi(3)…那一個音符。

   程式中以第 0 八度音爲基準頻率存於表格(BASE_FREQ)內。一旦讀到這筆資料後會根據低四位元查表搜尋到相對應的基底頻率，再乘上高四位元的 2 的冪次方，即爲所需送出的音符頻率。例如：

   5 → 45H(第 4 八度音的 So)　　→ $2^4 \times 98$=1568Hz

   7 → 27H(第 2 八度音的 Si) → $2^2 \times 123$=492Hz

   ※樂譜中有升半音及降半音，若高四位元的 MSB=1(Bit 7)表示該音符需要升(降)半音，至於是升或降則由低位數的 MSB(Bit 3)決定。當 MSB=1 表示升半音，MSB=0 表示降半音。

   $5^{\#}$ → 1 0 1 1 1 1 0 1

   $5^{\flat}$ → 1 0 1 1 0 1 0 1

   ※程式中以 00 表示樂曲結束。

2. 音符節拍：與拍子數有關，分別代表喇叭開啓與關閉的時間。在正常演奏速度下，一拍音符：

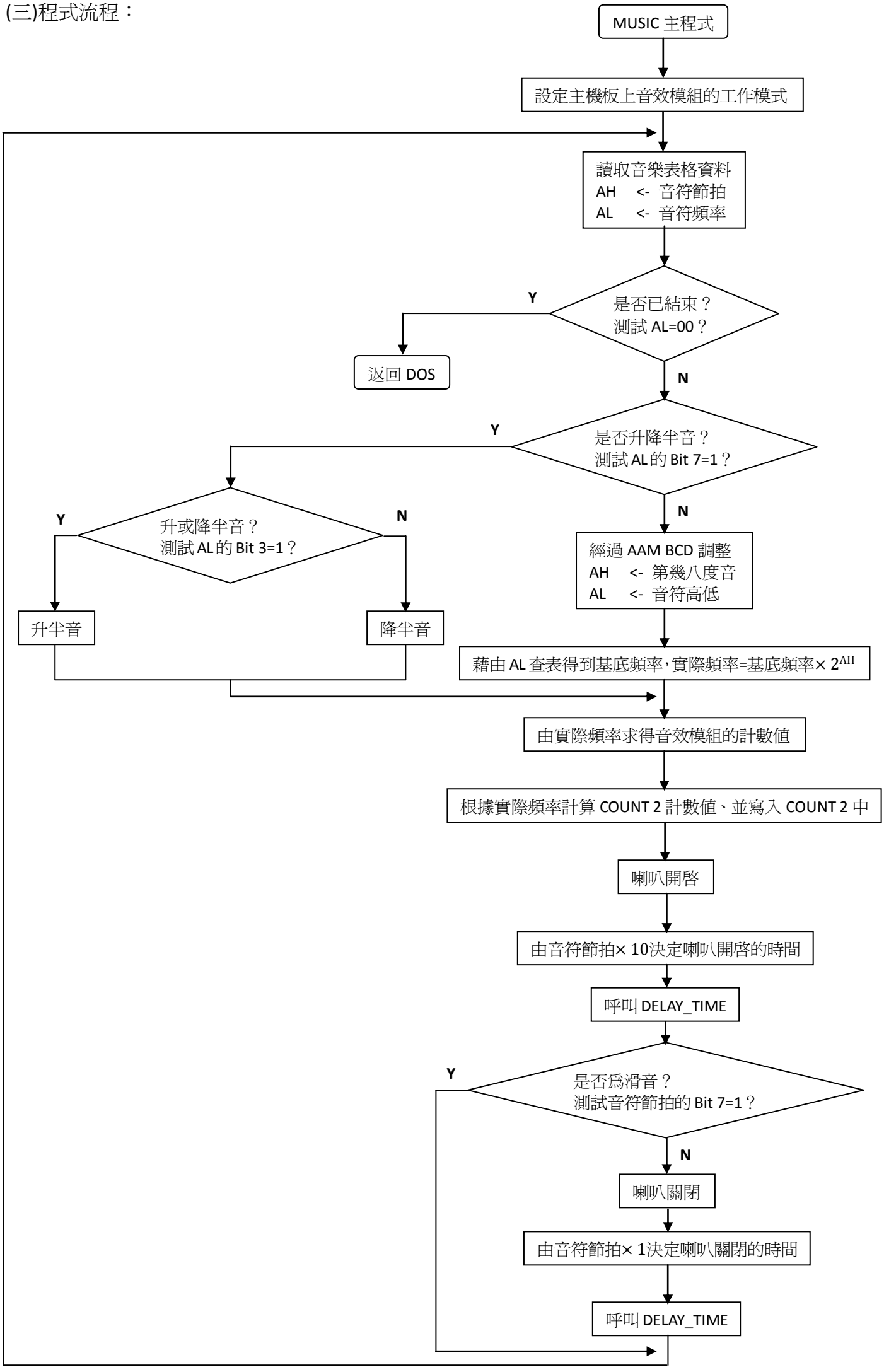
   Speaker ON 時間 = $0.44 \times \frac{10}{11} = 0.4$秒

   Speaker OFF 時間 = $0.44 \times \frac{1}{11} = 0.04$秒

   爲了延遲上述時間，程式中可呼叫 DELAY_TIME。要延遲 0.4 秒，則暫存器 AX 的值設爲 40；延遲 0.04 秒則設爲 4。爲了方便，代表音符節拍的位元組爲一拍時就將 AX 設爲 04、兩拍就設爲 08…依此類推。

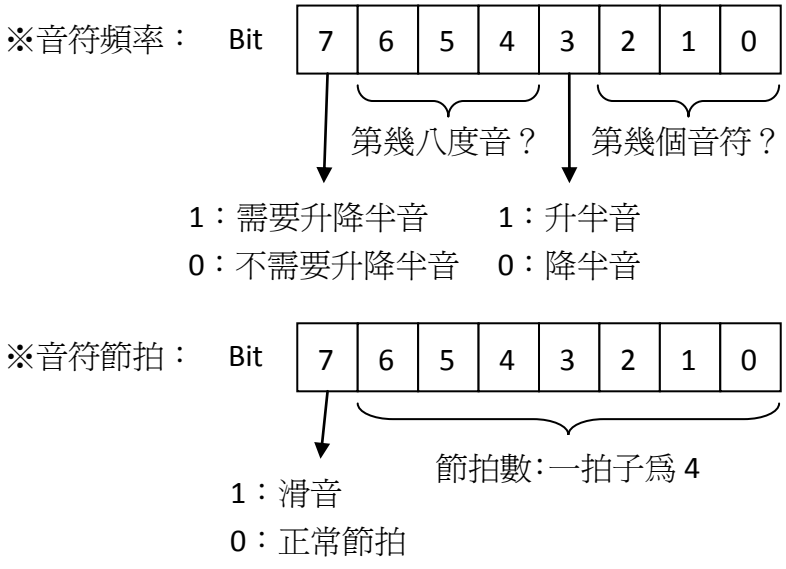
   ※若樂譜中出現滑音，表示兩音符間不中斷，也就是喇叭關閉的時間爲零。程式中定義將音符節拍這個位元組的 MSB 設爲 1 表示滑音。

(三)程式流程：


MUSIC 主程式
設定主機板上音效模組的工作模式
讀取音樂表格資料
AH <- 音符節拍
AL <- 音符頻率
是否已結束？
測試 AL=00？
Y
返回 DOS
N
是否升降半音？
測試 AL 的 Bit 7=1？
Y
N
升或降半音？
測試 AL 的 Bit 3=1？
Y
N
升半音
降半音
經過 AAM BCD 調整
AH <- 第幾八度音
AL <- 音符高低
藉由 AL 查表得到基底頻率，實際頻率=基底頻率× 2^AH
由實際頻率求得音效模組的計數值
根據實際頻率計算 COUNT 2 計數值、並寫入 COUNT 2 中
喇叭開啓
由音符節拍× 10決定喇叭開啓的時間
呼叫 DELAY_TIME
是否爲滑音？
測試音符節拍的 Bit 7=1？
Y
N
喇叭關閉
由音符節拍× 1決定喇叭關閉的時間
呼叫 DELAY_TIME

(四)程式說明：

1. 首先觀察 146~168 行的資料段，包括：
   (1) 基底頻率(BASE_FREQ)：Do、Re、Mi…Si 等七個音符的第 0 八度音資料。
   (2) 樂曲表格資料(MUSIC TABLE)：每個音符的表示法如下圖所示。



2. 這個程式第一部份是利用 LODSW 指令讀取音樂表格資料(11~17 行)，並將此音符的頻率與節拍分別置於 AL 與 AH 暫存器內。接著根據 AL 資料作幾項判斷：
   (1) 若 AL=00？ → 判斷樂曲是否結束？(19~20 行)
   (2) 若 AL 的 Bit 7=0？ → 正常音符的處理。(26~32 行)

   執行 NORM_SOUND：

   ※將 AL 原始資料利用 AAM 指令拆成兩個 BCD → 此時 AH=第幾八度音，AL=第幾個音符。

   ※利用 XLAT 指令，以 BASE_FREQ 爲基底位址，第幾個音符爲偏移量，查表得到目前音符中第 0 八度音的頻率，重新置於 AL 中。

   ※計算音符的頻率 → $\mathrm{AX} = \mathrm{AL} \times 2^{\mathrm{AH}}$

   (3) 若 AL 的 Bit 7=1？ → 需要升半音或降半音。(23~24 行)

   執行 U_D_HALF：(34~41 行)

   ※將 Bit 7 重置爲 0。

   ※將 AL 原始資料利用 ROR 指令拆成兩段。

   ※判斷 AL 的 Bit 3=1？ → 升半音。

   執行 U_HALF：(50~53 行)

   ◎將 Bit 3 重置爲 0。

   ◎將原來音符儲存至 CL、並將原來音符遞增一個音階。

   ※判斷 AL 的 Bit 3=0？ → 降半音。

   執行 D_HALF：(44~48 行)

   ◎將 Bit 3 重置爲 0。

   ◎將原來音符儲存至 CL、並將原來音符遞減一個音階。

   ※執行升、降半音的處理。

   執行 HALF_PROCESS：(55~67 行)

   ◎利用 XLAT 指令，分別查表得到原來音符與遞增或遞減後音符的第 0 八度音的頻率(分別在 CL 與 AL)。

   ◎計算音符的頻率 → $\mathrm{AX} = (\mathrm{AL} + \mathrm{CL})/2 \times 2^{\mathrm{AH}}$

3. 這個程式的第二部份在處理音符節拍的問題，過程如下：
   (1) 開啓喇叭。(79~81 行)。
   (2) 取出音符節拍數乘上 10，呼叫 DELAY_TIME 便是喇叭開啓的時間。(83~89 行)

(3) 判斷是否爲滑音？(90~91 行)

(4) 關閉喇叭。(93~95 行)

(5) 取出音符節拍數，呼叫 DELAY_TIME 便是喇叭關閉的時間。(97~99 行)

(五)心得

藉著這次的作業，我練習使用了組合語言來做簡單的硬體控制來發出聲音，並調整頻率使其能組成一首完整的音樂。由於對指令的了解不夠多、不夠深，沒辦法做出預期中的效果，如播放速度的調整、停止、讀取其他檔案的樂譜、選取歌曲等等，這是我應該要繼續學習的。組合語言使用在中斷程式、常駐程式及 I/O 的控制上相當方便，也因此我必須對組合語言的使用更加熟悉，以作爲了解電腦運作更良好的基礎。

MUSIC.ASM

```
1   TIMER_2      EQU 42H
2   TIMER_CTRL   EQU 43H
3   PPI_B        EQU 61H
4
5   .MODEL  SMALL
6   .CODE
7
8   MOV      AL,0B6H
9   OUT TIMER_CTRL,AL
10
11  MOV      AX,@DATA
12  MOV      DS,AX
13  MOV      SI,OFFSET MUSIC_TABLE
14  MOV      BX,OFFSET BASE_FREQ
15
16  AGAIN:
17  LODSW
18
19  CMP AL,00
20  JE   E   ND_MUSIC
21  PUSH     AX
22
23  TEST AL,80H
24  JNZ      U_D_HALF
25
26  NORM_SOUND:
27  AAM
28  XLAT
29  MOV      CL,AH
30  MOV      AH,00
31  SHL      AX,CL
32  JMP      TIMER_OUT
33
34  U_D_HALF:
35  MOV      AH,AL
36  AND      AL,0FH
37  AND      AH,0F0H
38  MOV      CL,4
39  ROR      AH,CL
40  AND      AH,0F7H
41  TEST     AL,08H
42  JNZ      U_HALF
43
```

```
44  D_HALF:
45  AND AL,0F7H
46  MOV     CL,AL
47  DEC     AL
48  JMP     HALF_PROCESS
49
50  U_HALF:
51  AND     AL,0F7H
52  MOV     CL,AL
53  INC     AL
54
55  HALF_PROCESS:
56  PUSH    AX
57  MOV     AL,CL
58  XLAT
59  MOV     CL,AL
60  POP     AX
61  XLAT
62  CLC
63  ADD     AL,CL
64  RCR     AL,1
65  MOV     CL,AH
66  MOV     AH,00
67  SHL     AX,CL
68
69  TIMER_OUT:
70  MOV     CX,AX
71  MOV     DX,18
72  MOV     AX,10352
73  DIV     CX
74
75  OUT     TIMER_2,AL
76  MOV     AL,AH
77  OUT     TIMER_2,AL
78
79  IN      AL,PPI_B  ;OPEN SPEAKER
80  OR      AL,3
81  OUT     PPI_B,AL
82
83  POP     AX
84  MOV     CL,AH
85  AND     AH,7FH
86  MOV     AL,AH
87  MOV     AH,10
88  MUL     AH
```

```
89  CALL    DELAY_TIME
90  TEST    CL,80H
91  JNZ     AGAIN
92
93  IN      AL,PPI_B  ;CLOSE SPEAKER
94  AND     AL,0FCH
95  OUT     PPI_B,AL
96
97  MOV     AL,CL
98  MOV     AH,00H
99  CALL    DELAY_TIME
100
101 JMP     AGAIN
102
103 END_MUSIC:
104 MOV     AH,4CH
105 INT     21H
106
107 DELAY_TIME  PROC    NEAR
108 PUSH    BX
109 PUSH    CX
110 PUSH    DX
111 MOV     BX,AX
112
113 MOV     CH,00H
114 MOV     CL,00H
115 MOV     DH,00H
116 MOV     DL,00H
117 MOV     AH,2DH
118 INT     21H
119 TEST_LOOP:
120 MOV     AH,2CH
121 INT     21H
122
123 PUSH    DX
124 MOV     AX,6000
125 MUL     CX
126 POP     DX
127 PUSH    AX
128 MOV     AL,100
129 MUL     DH
130 MOV     CX,AX
131 POP     AX
132 ADD     AX,CX
133 MOV     DH,00
```

```
134 ADD       AX,DX
135
136 CMP       AX,BX
137 JL        TEST_LOOP
138
139 POP       DX
140 POP       CX
141 POP       BX
142 RET
143 DELAY_TIME    ENDP
144
145 .DATA
146 BASE_FREQ     DB   255,65,73,82,87,98,110,123
147 MUSIC_TABLE DB   70,6,34,3,35,3,36,6,36,3,41,3,42,6,42,3,46,3,45,3,44,3,44,3,42,3,41,12
148         DB   42,3,41,3,41,3,36,3,35,3,34,3,32,3,34,3,35,18
149         DB   34,3,35,3,36,6,36,3,41,3,42,6,42,3,46,3,45,3,44,3,44,3,45,3,44,12
150         DB   41,6,41,3,36,3,43,3,44,3,34,3,35,3,36,12,35,9,34,3,34,24
151         DB   70,6,36,6,36,3,35,3,36,6,70,6,36,3,41,3,36,3,35,3,36,12
152         DB   34,6,34,3,35,3,36,3,41,3,36,6,35,6,35,3,34,3,35,6,70,6
153         DB   36,6,41,3,36,3,42,3,41,9,42,3,41,3,41,3,36,3,41,9
154         DB   31,3,36,6,35,3,36,3,41,6,36,3,35,3,35,6,35,3,34,3,35,6
155         DB   34,3,41,3,36,6,36,3,35,3,36,6,70,6,36,3,41,3,36,3,35,3,36,12
156         DB   34,6,34,3,35,3,36,3,41,3,36,6,35,6,35,3,34,3,35,6,70,6
157         DB   36,6,41,3,36,3,42,3,41,9,42,3,41,3,41,3,36,3,41,12
158         DB   36,6,35,3,36,3,41,6,36,3,35,3,35,3,34,18,70,3
159         DB   34,3,35,3,36,6,36,3,41,3,42,6,42,3,46,3,45,3,44,3,44,3,42,3,41,12
160         DB   42,3,41,3,36,3,35,3,34,6,32,3,34,3,35,6,35,3,34,3,35,6
161         DB   34,3,35,3,36,6,36,3,41,3,42,6,42,3,46,3,45,3,44,3,44,3,45,3,44,12
162         DB   41,6,41,3,36,3,43,3,44,3,34,3,35,3,36,12,35,12
163         DB   34,3,31,3,34,3,35,3,36,3,41,3,42,3,44,3,45,9,46,3,44,6
164         DB   45,134,45,3,41,3,44,3,45,3,46,3,51,3,45,3,46,3,44,24
165         DB   70,3,31,3,34,3,35,3,36,3,41,3,42,3,44,3,45,9,46,3,44,134
166         DB   45,3,41,3,44,3,45,3,46,3,51,3,45,3,46,3,44,12,00
167
168 .STACK
169         END
```