

《微机原理与汇编语言》

课程报告

|  |  |
| --- | --- |
| 学 院： | 计算机工程学院 |
| 专　　业： | 网络工程 |
| 执行学期： | 2024-2025-1 |
| 班 　级： | 22网络1 |
| 学 号： | 2022146101 |
| 姓 名： | 齐智暄 |
|  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 任务1 | | | 任务2 | |  |
|  | 课程总结 | 课程设计 | 文献阅读 | 原理图 | 程序 | 总成绩 |
| **成绩** |  |  |  |  |  |  |
| **批阅教师** |  |  |  |  |  |  |

2024年12月6日

对基于8254的电子键盘乐器演奏程序系统的设计

1. **课程总结**

随着科技的不断进步，计算机科学与技术领域也在迅速演变，推动着各行各业的数字化转型。在这个时代，理解计算机的微机原理以及掌握接口技术成为计算机科学专业学生不可或缺的基础。

首先，微机原理作为课程的核心，为学生提供了深入了解计算机内部结构和运作原理的机会。在当今计算机科学领域，硬件与软件的协同工作至关重要。微机原理的学习使学生能够理解计算机的底层机制，有助于他们更好地优化软件性能，设计更高效的算法，并更好地解决实际问题。

其次，接口技术是当前科技进步中的热点之一。随着物联网、云计算和大数据的兴起，设备之间的连接和通信变得愈发复杂。《微机原理与接口技术》课程培养学生设计各种接口电路的能力，使他们能够适应快速发展的技术环境。这对于从事嵌入式系统、物联网设备开发等领域的专业人才尤为关键。

此外，当前计算机科学技术的发展趋势还包括人工智能、机器学习等前沿技术的广泛应用。微机原理的深入学习为理解和优化这些复杂算法提供了基础，而接口技术的应用也在人机交互、智能设备控制等方面发挥着关键作用。因此，该课程使学生能够更好地参与和推动这一潮流。

另外，安全性与隐私保护是当前计算机科学领域的重要议题。微机原理的学习使学生能够理解计算机系统的脆弱性，而接口技术的应用则需要关注数据传输的安全性。这反映了当前社会对于数字安全的不断增强的需求，也使得学生在未来能够成为更具责任感的计算机专业人士。

**《微机原理与接口技术》课程培养主要包括以下几个方面：**

1.理论基础： 掌握计算机体系结构、微机原理和接口技术的基本理论，深入了解计算机硬件组成和工作原理。

2.实际操作能力： 具备微机系统设计、调试、接口编程等实际操作的能力，能够熟练使用汇编语言和高级语言进行编程。

3.接口设计： 能够设计各种接口电路，理解外设与微机之间的通信原理，提高硬件与软件协同工作的能力。

4.问题解决： 具备分析和解决微机系统出现问题的能力，提高对计算机系统整体性能的优化水平。

**微机原理与接口技术的主要技术包括：**

1.计算机体系结构： 深入了解计算机的层次结构，包括硬件和软件的相互关系。

2.微机原理： 学习微处理器的结构、指令系统、寻址方式等基本原理，了解微机系统的整体设计。

3.接口技术： 理解不同设备之间的接口标准和通信协议，学习如何设计合理的接口电路。

4.汇编语言编程： 掌握汇编语言的基本语法和编程技巧，能够进行底层程序设计。

5.系统调试与优化： 学习微机系统的调试方法和性能优化策略，提高系统的稳定性和效率。

**微机原理与接口技术是计算机专业中的重要课程，与其他课程有着紧密的关系，包括：**

1.计算机组成原理： 提供计算机硬件组成和运行原理的基础知识。

2.数字电路与系统设计： 强化数字电路和系统设计的概念，为接口电路的设计打下基础。

3.嵌入式系统设计： 进一步深化嵌入式系统的开发和设计理念，拓展硬件与软件的融合能力。

**在当前计算机科学技术发展的背景下，人工智能、物联网、云计算等领域的迅速崛起对微机原理与接口技术提出了新的要求：**

1.人工智能与边缘计算： 微机原理与接口技术需要更好地适应人工智能和边缘计算的需求，提高对各种传感器和执行器的接口设计水平。

2.云计算： 理解云计算基础设施的原理，能够进行云端与终端的通信与数据传输。

3.物联网： 加深对物联网协议和通信方式的理解，提高对设备接口和通信协议的设计水平。

**二、课程系统设计**

设计一个可以通过微机键盘演奏不同音乐的控制系统，系统通过按下微机键盘模仿电子琴键驱动实验箱上的喇叭发生，实现演奏音乐的功能。

1. **基本功能要求**

能够演奏 C 调包含高、中、低音的不同节拍的乐曲，音调与频率的对应关系如下：

|  |  |
| --- | --- |
| 中 音 | 1-523Hz, 2-575Hz, 3-659Hz, 4-698Hz, 5-784Hz, 6-880Hz, 7-988Hz |
| 低 8 度音 | 基本音频率/2 例如低音1为 523/2=261.5Hz |
| 高 8 度音 | 基本音频率\*2，例如高音1为 523\*2=1046Hz |

微机键盘与电子琴键盘的对应关系为：高音依次为 Q,W,E,R,T,Y,U，中音依次为A,S,D,F,G,H,J，低音依次为 Z,X,C,V,B,N,M。

自动演奏乐曲功能。当按下 P 键后，自动播放一首乐曲，曲长大于 20 秒。

琴键图形显示功能。屏幕上显示 21 个琴键（高中低音），当按下某键后屏幕上的对应琴键有所反映。

一个音调 D，当从 C 调改为 D 调后，演奏音按照 D 调(比 C 高一个调)发音。

1. **设计思路**

一首乐曲由若干音符组成，一个音符对应一个频率，将与一个频率对应的计数初值写入计数器就可以产生相应的频率，计算公式如下：

当音符对应的频率确定后，还需要知道每个音符演奏的时间。我们知道，音符的演奏时间是由节拍控制的，分为一拍，半拍， 拍， ……，如果在一首乐曲中，音符演奏的最短时间是 拍，我们就将 拍作为一个最短时间单位 1，那么 拍单位时间就是 2， 拍就是 4，一拍就是 8，假定一拍的时间为 1s,那么 ,, 拍的时间就是0.5s,0.25s,0.125s。

编程首先应该对计数器初始化，然后扫描键盘，根据扫描结果选择对应的频率，输出到实验箱的音频电路输入端口，并调用延时子程序控制节拍。

1. **硬件设计方案及硬件连线图**

**a)设计方案**

我们为了控制音调的高低，使用的是8254定时器，使用方式三，通过改变分频值来控制。分频值的计算是用初始的时钟频率32.768KHZ来除以想得到的频率，然后舍入成整数来作为分频值输入8254。  
具体的连线情况大致如上图，从总线上的LD0-LD7接8254的D0-D7，LA0和LA1接8254的A0和A1，IOWR和IORD接8254的WR和RD，地址0000H接8254的CS。除此之外，8254的GATE0接VCC，CLK0接初始的时钟频率32.768KHZ，OUT0接音频电路即可。

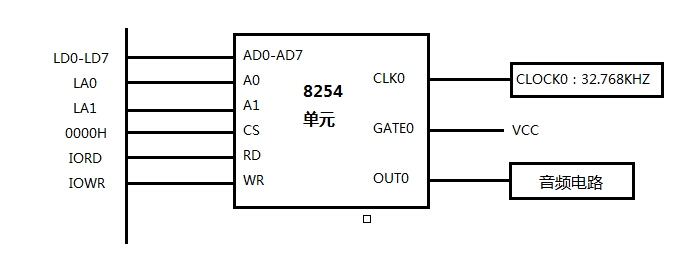


图 1：接线示意图

**b)控制字计算**

我们选择使用计数器0的工作方式3进行分频，计数初值只需8位，因此读/写低八位，计数初值的形式是选择十进制BCD码计数。

****

图 2：8254控制字的格式

****

图 3：8254读出控制字的格式

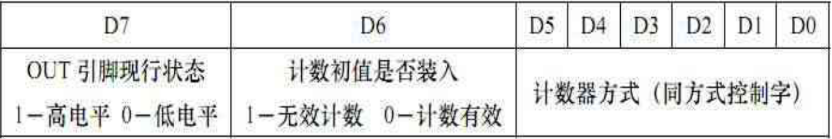
****

图 4：8254状态字格式

**c)计数初值的计算**

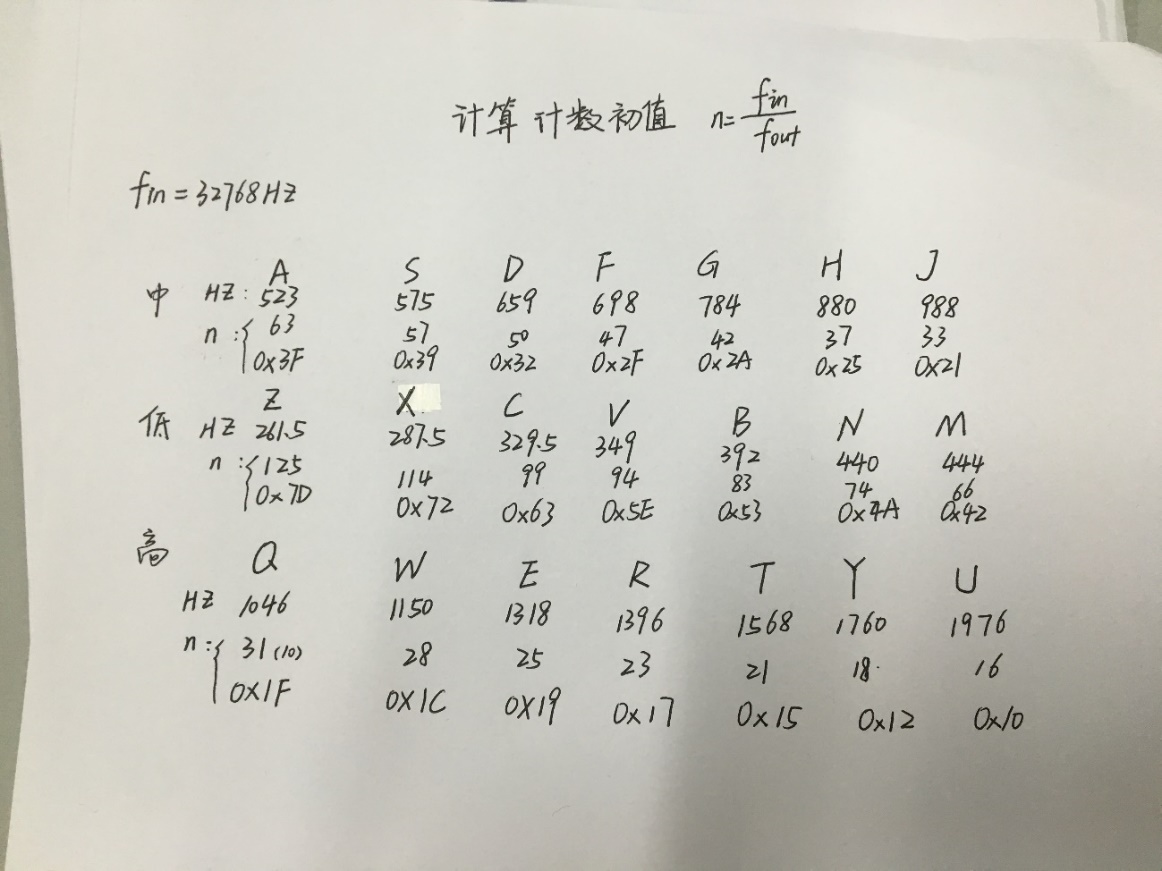
****

图 5：计数初值的计算

1. **软件设计方案**

软件工作流程示意图如下：

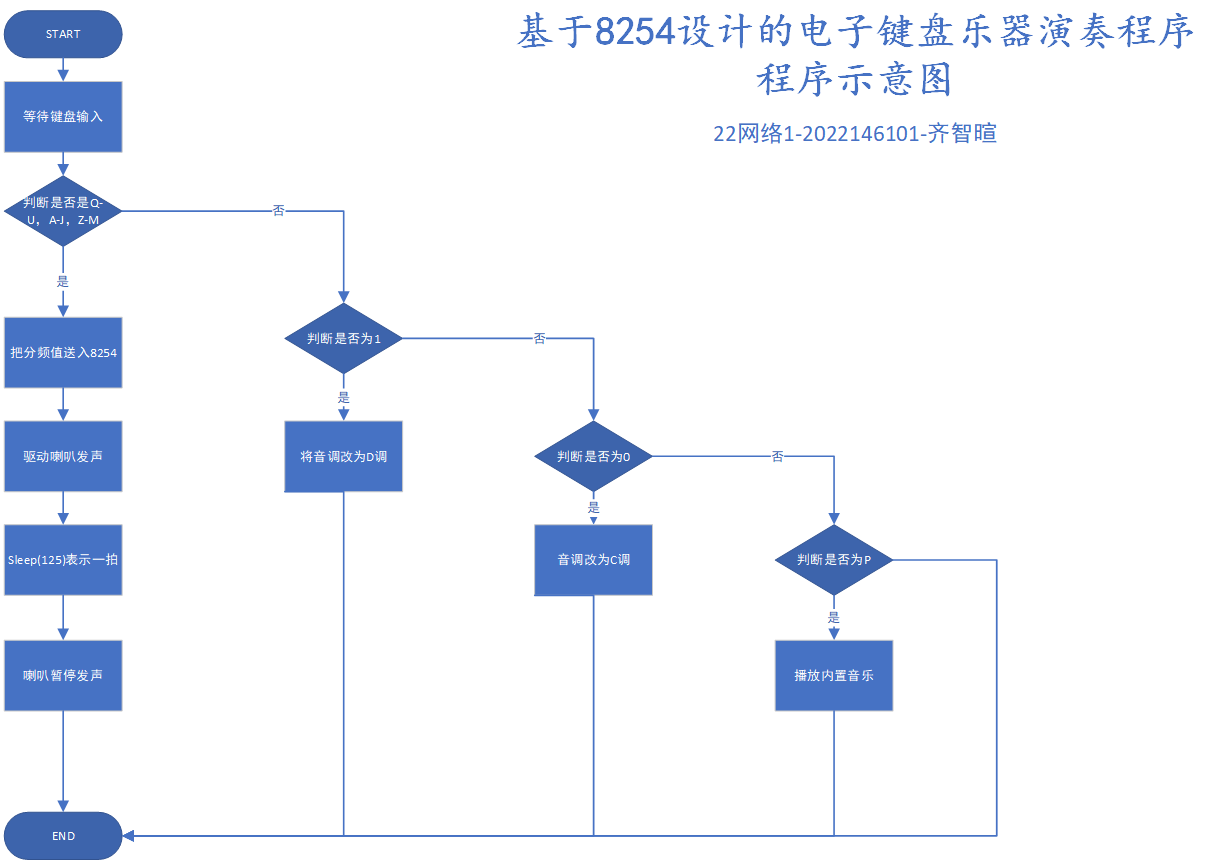
****

图 6：程序示意图

**自动播放音乐的实现:**

对于一首音乐：我们将其分为了三个部分：

Music[]:音符，用1234567代替do,re,mi,fa,suo,la,xi;

Musicc[]:各音符之间的延时，可以从乐谱里面看到；

Musiccc[]:各音符的调，分别用2，1，0高中低音调；

分别依据乐谱谱写我们所需要的数组，然后通过循环遍历来调用函数来给硬件传递分频的计数值，最终能播放整首音乐。

**高低音的实现：**

初始化时，将音调全部置为中调，即为1，对于高音，只需要将musiccc[i]++即可，同理低音就musiccc[i]--;

**每个音符的初始化及键盘键入设置：**

**音符的频率是我们算出来的，然后在实际调试过程中，我们稍微改变了一点，**使我们的电子琴更接近原生的音乐，例如高音的do，用键盘上的A键触发，十六进制为0x41,do对应的频率为0x003c，使用这样一句进行初始化：

键盘键入使就触发OnButton（）函数，相当于用鼠标控制软件一样。

将音乐的节奏和音调存储在数组里面，遍历数组，检测到对应字符调用相应的函数控制扬声器发出频率（音调）不同的声音。

**具体程序实现：**

DATA SEGMENT

; 键码变量

KEYCODE DB ? ; 当前按键的键码

FREQ\_TABLE\_HIGH DW 31, 28, 25, 24, 21, 19, 16 ; 高音分频值表

FREQ\_TABLE\_MID DW 63, 56, 50, 47, 42, 37, 31 ; 中音分频值表

FREQ\_TABLE\_LOW DW 126, 112, 100, 94, 84, 74, 63 ; 低音分频值表

CURRENT\_TABLE DW 0 ; 当前音调分频表指针

; 自动演奏数据

MUSIC\_NOTES DB 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1 ; 音符序列

MUSIC\_DELAY DB 8, 8, 8, 8, 8, 16, 8, 8, 8, 8, 8 ; 节拍时间序列

NOTE\_COUNT DB 11 ; 音符数量

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START:

; 初始化数据段

MOV AX, DATA

MOV DS, AX

; 初始化8254

CALL INIT\_8254

; 设置默认音调表为中音

MOV AX, OFFSET FREQ\_TABLE\_MID

MOV CURRENT\_TABLE, AX

MAIN\_LOOP:

; 扫描键盘

CALL SCAN\_KEYBOARD

; 检查特殊按键功能

CMP KEYCODE, 'P' ; 自动演奏

JE AUTO\_PLAY

CMP KEYCODE, '1' ; 切换高音

JE SHIFT\_HIGH

CMP KEYCODE, '2' ; 切换低音

JE SHIFT\_LOW

; 检查普通按键（高音Q~U, 中音A~J, 低音Z~M）

CALL MAP\_KEY\_TO\_FREQ

CALL PLAY\_NOTE

JMP MAIN\_LOOP

; 初始化8254计数器

INIT\_8254 PROC

MOV AL, 36H ; 设置8254计数器为工作方式3，低高字节模式

OUT 43H, AL ; 写入控制字寄存器

RET

INIT\_8254 ENDP

; 键盘扫描函数

SCAN\_KEYBOARD PROC

IN AL, 60H ; 从键盘端口读取键码

MOV KEYCODE, AL ; 保存键码

RET

SCAN\_KEYBOARD ENDP

; 按键映射到分频值

MAP\_KEY\_TO\_FREQ PROC

MOV BX, CURRENT\_TABLE ; 获取当前音调表

; 检测按键并加载对应频率值

CMP KEYCODE, 'Q' ; 高音1

JE LOAD\_FREQ\_1

CMP KEYCODE, 'W' ; 高音2

JE LOAD\_FREQ\_2

CMP KEYCODE, 'A' ; 中音1

JE LOAD\_FREQ\_3

CMP KEYCODE, 'S' ; 中音2

JE LOAD\_FREQ\_4

CMP KEYCODE, 'Z' ; 低音1

JE LOAD\_FREQ\_5

LOAD\_FREQ\_1:

MOV AX, [BX]

JMP SET\_FREQ

LOAD\_FREQ\_2:

MOV AX, [BX+2]

JMP SET\_FREQ

LOAD\_FREQ\_3:

MOV AX, [BX+4]

JMP SET\_FREQ

LOAD\_FREQ\_4:

MOV AX, [BX+6]

JMP SET\_FREQ

LOAD\_FREQ\_5:

MOV AX, [BX+8]

SET\_FREQ:

RET

MAP\_KEY\_TO\_FREQ ENDP

; 播放单个音符

PLAY\_NOTE PROC

; 设置分频值到8254

MOV AL, AH

OUT 40H, AL ; 写高字节

MOV AL, AL

OUT 40H, AL ; 写低字节

; 延时控制音符长度

CALL DELAY

RET

PLAY\_NOTE ENDP

; 自动演奏

AUTO\_PLAY PROC

MOV CX, NOTE\_COUNT ; 获取音符数量

MOV SI, OFFSET MUSIC\_NOTES

MOV DI, OFFSET MUSIC\_DELAY

AUTO\_PLAY\_LOOP:

MOV AL, [SI] ; 加载当前音符

CALL MAP\_KEY\_TO\_FREQ

CALL PLAY\_NOTE

INC SI ; 指向下一个音符

INC DI ; 指向下一个延时时间

LOOP AUTO\_PLAY\_LOOP

RET

AUTO\_PLAY ENDP

; 切换高音

SHIFT\_HIGH PROC

MOV AX, OFFSET FREQ\_TABLE\_HIGH

MOV CURRENT\_TABLE, AX

RET

SHIFT\_HIGH ENDP

; 切换低音

SHIFT\_LOW PROC

MOV AX, OFFSET FREQ\_TABLE\_LOW

MOV CURRENT\_TABLE, AX

RET

SHIFT\_LOW ENDP

; 延时子程序

DELAY PROC

MOV CX, 0FFFFH ; 设置延时循环

DELAY\_LOOP:

LOOP DELAY\_LOOP

RET

DELAY ENDP

CODE ENDS

END START

**参考文献**

[1]

[2]

**文献阅读报告：**

**综合应用：**