



中南大學
CENTRAL SOUTH UNIVERSITY

算法与分析设计

题 目： 算法实验

学生姓名： 陈康坤

指导教师： 李敏

学 院： 计算机学院

专业班级： 计科 2202

2024 年 5 月 27日

目录

1. 问题描述

- 1) 基本..... 2
- 2) 拓展功能..... 2

2. 系统设计

- 1) 总体架构..... 3
- 2) 数据段定义..... 4
- 3) 宏..... 5
- 4) 入口程序..... 6
- 5) 子程序..... 7

3. 源代码清单

- 1) 子程序功能及参数列表..... 13
- 2) 调用关系..... 14

4. 运行结果测试与分析

- 1) 运行结果测试..... 16

5. 结论与心得

- 1) 程序调试中发现的问题和解决办法..... 19
- 2) 可能但因时间关系没有来得及完成的想法..... 19
- 3) 实验心得..... 20

问题描述

1. 基本要求

本实验基于计算机原理和汇编语言实现了一个简易的电子琴模拟系统，其核心功能如下：

- 1) 按下按键 1-8 时，依次发出 8 个音调；
- 2) 按 CTRL+C 退出程序；
- 3) 按键的音调频率表（524、588、660、698、784、880、988、1048）。

2. 拓展功能

除实现基本需求之外，本实验拟实现如下拓展功能：

- 1) 音域拓展：按下键盘中特定的 21 个键，可以发出对应的 21 个音符。按下其他键时会弹出错误提示，帮助用户弹奏更丰富的音乐；
- 2) 演奏记录：用户每演奏完一次，系统将用户的演奏记录写入到文件中，后续可选择读取并播放演奏记录，帮助用户提升水平；
- 3) 休止符打印：用户不弹奏时，每隔 0.5s 打印一个休止符，帮助用户控制弹奏节奏；
- 4) 新手教程：通过显示键盘与音符的对应关系，播放一段预置音乐和实时显示预置音乐对应的音符，帮助用户快速上手。

通过加入这些功能，本实验不仅实现了基本的电子琴模拟，还提供了扩展的音域和其他辅助工具，帮助用户更好地学习和演奏。

系统设计

1. 总体架构

系统设计遵循模块化原则，主要分为以下几个核心模块：

- 1) 数据段：定义了系统运行过程中所需的各种字符串和数据。例如音符、菜单和休止符等，确保系统在运行时能够访问和使用这些预定义的数据；
- 2) 宏：提供了一个用于打印数据的宏，简化了后续程序中的打印操作，使代码更加简洁和易于维护；
- 3) 入口程序：负责接收用户输入的参数，并将这些参数与对应的操作进行映射。根据用户的选择，入口程序调用相应的子程序来执行特定的功能，执行完毕后循环显示菜单以供后续操作；
- 4) 子程序：本系统包含多个子程序以提高代码的重用性和模块化程度，每个子程序都实现了特定的功能。调用子程序时，只需传入相应的参数即可。

下面对这些模块的具体实现进行逐步具体说明。（注：下文所涉及图片均为代码片段而非完整代码）

2. 数据段定义

A. 键盘与音符映射

- 1) 键盘真实映射：“`keyboard_true`”定义了键盘上特定按键与音符的映射。例如，“1234567qwertyuasdfghj”这些键对应特定的音符。
- 2) 键盘扩展映射：“`keyboard_change`”定义了键盘上更多按键与音符的映射，使得键盘可以发出更丰富的音符。
- 3) 音符变换映射：“`note_change`”和“`note_change2`”定义了按键对应的音符名称和音阶，帮助用户识别不同按键发出的音符。

B. 菜单和信息提示

- 1) 菜单：`menu` 定义了系统的主菜单，包括选项和提示信息，用于用户操作选择；
- 2) 输入缓冲区：`content` 用于存放用户输入的按键 ASCII 码；
- 3) 文件名：`filename` 定义了导出历史记录文件的文件名；
- 4) 文件句柄：`handle` 用于存放打开文件的句柄；

- 5) 提示信息: **mess1**, **mess2** 和 **mess3** 是各种提示信息, 分别用于不同的操作场景, 例如提示用户开始演奏、当前正在演奏、以及键盘和音符的对应关系。

C. 频率和时长数据

- 1) 频率数据: **freq** 定义了一系列频率值, 用于生成特定节奏的音乐;
- 2) 时间数据: **time** 定义了对应频率的持续时间, 配合 **freq** 数据用于生成带有节奏的音乐。

D. 预置音符和结果存储

- 1) 预置音符: **note1** 和 **note2** 定义了一段预置的音符序列, 用于播放示例音乐或帮助用户快速上手;
- 2) 结果存储: **result** 和 **res** 用于存放计算结果或临时数据, 便于后续操作使用。

```
music_table dw 131, 147, 165, 175, 196, 220, 247
             dw 262, 294, 330, 349, 392, 440, 494
             dw 523, 587, 659, 698, 784, 880, 988

keyboard_true db "1234567qwertyuasdfghj$"
keyboard_change db "123456789:;<=>?@ABCDEFGHIJKLMNPOQRSTUVWXYZ"
note_change db "CDEFGABCDEFGFGAB"
note_change2 db "333333344444445555555"
menu db 10,13,10,13,'*** MENU ***'
      db 10,13
      db 10,13,' 1. Play'
      db 10,13,' 2. History'
      db 10,13,' 3. Tutor'
      db 10,13,' 0. Quit'
      db 10,13
      db 10,13,'please choose one of 0-3:', '$'
content db 100 dup(0) ;存放输入按键的ASCII码
filename db 'history',0 ;导出文件的文件名
handle dw ? ;导出文件的文件句柄
mess1 db 10,13,'please start your performance,exit with ctrl+c', '$'
mess2 db 10,13,'Playing now.....', '$'
mess3 db 10,13,'*****'
      db 10,13,' 1 2 3 4 5 6 7 '
      db 10,13,' C3 D3 E3 F3 G3 A3 B3 '
      db 10,13,' '
      db 10,13,' Q W E R T Y U '
      db 10,13,' C4 D4 E4 F4 G4 A4 B4 '
      db 10,13,' '
      db 10,13,' A S D F G H J '
      db 10,13,' C5 D5 E5 F5 G5 A5 B5 '
      db 10,13,'*****', '$'
```

Fig1. 数据段定义

3. 宏

A. “show” 宏

- 1) 功能: 显示字符串;

- 2) 入口参数：字符串标号；
- 3) 出口参数：无；
- 4) 步骤：将字符串标号加载到 DX 寄存器中，设置 AH 寄存器为 9，调用 DOS 中断 21H 的 9 功能号，用于显示字符串，最后调用 DOS 中断 21h 显示字符串。

B. “find” 宏

- 1) 功能：在字符串中查找字符；
- 2) 入口参数：char、str、result；
- 3) 出口参数：无；
- 4) 步骤：初始化字符位置为 -1，将字符串起始地址加载到 SI 寄存器中。增加 CX 寄存器的值表示当前字符的位置，从字符串中读取一个字符到 AL 寄存器中，SI 自动增加 1，检查是否到达字符串末尾。若是，则跳转到 notFound 标签，否则比较当前字符与目标字符，若相等，则跳转到 foundChar 标签，否则继续查找下一个字符。

```
; 此时CX寄存器中储存的是字符在字符串中的位置 (0-based)
;*****
; 功能：显示字符串
; 入口参数：字符串标号
; 出口参数：无
;*****
show macro _Label
    lea dx, _Label ;字符串标号
    mov ah,9
    int 21h
endm
;*****
;功能：在字符串中查找字符
;入口参数：char, str, result
;出口参数：无
;*****
find macro char, str, result
    LOCAL notFound
    mov cx, -1 ; 初始化字符位置为-1
    mov si, offset str ; 指向字符串开始
searchChar:
    inc cx
    lodsb ; 从字符串中读取一个字符到AL
    cmp al, '$' ; 检查是否到达字符串末尾
    je notFound ; 如果是，则说明字符未找到
    cmp al, char ; 比较当前字符与目标字符
    je foundChar ; 如果相等，则找到字符
    jmp searchChar ; 继续搜索下一个字符
foundChar:
    dec si ; 因为lodsb会使si加1，所以需要减回来以得到正确的位置
    mov result, cx
    jmp oo
notFound:
    mov result, -1
```

Fig2. 宏

4. 入口程序

A. 初始化

- 1) 加载数据段：将数据段的起始地址加载到 DS 寄存器中，以便访问数据段中的数据；
- 2) 显示菜单：调用前面编写的“show”宏和“menu”数据段，使用 show menu 指令显示菜单内容。

B. 读取用户输入

- 1) 检查输入的有效性：使用 21h 中断的 1 号功能从键盘读取一个字符，该字符的 ASCII 码保存在 AL 寄存器中，同时判断用户输入的字符是否在合理的范围内（'0' 到 '3'）。若输入字符小于 '0' 或大于 '3'，则跳至“no”标签；
- 2) 处理有效输入：将输入的 ASCII 码转换为相应的数字，即将字符 '0' 到 '3' 转换为数字 0 到 3，使用转换后的数字作为索引，调用相应的子程序，实现对应的功能；
- 3) 无效输入处理：如果用户输入的字符不在合理范围内，即无效输入，则程序会跳转到 no 标签处，重新显示菜单，等待用户重新输入。

```
start:
    mov ax,@data
    mov ds,ax
disp:
    show menu
    mov ah,1
    int 21h           ;调用21h中断的第1号功能，从键盘读入字符，AL保存读入字符的ASCII码
    cmp al,30h
    jb no
    cmp al,33h
    ja no
    sub al,30h        ;将输入的ASCII码转为BCD码
    mov bl,al
    mov bh,0
    add bx,bx         ;数字乘2才是直接定址表中的标号的地址
    call word ptr fun_table[bx] ;调用子程序
    jmp disp          ;重复显示菜单
no:
```

Fig3. 入口程序

5. 子程序

A. “Play” 子程序

- 1) 显示信息：使用 show mess1 指令显示提示信息，告知用户可以开始弹奏钢琴，并使用中断 21h 的功能显示换行符；

- 2) 创建历史记录文件：使用中断 21h 的功能，根据数据段指定的文件名创建一个新文件，并保存文件句柄起来；
- 3) 打印休止符：使用中断 15h 的功能，设置时钟中断频率为每秒 2 次，以实现约 0.5 秒的延迟效果。同时循环读取键盘输入，使用中断 16h 的功能检测键盘输入，如果有输入则跳过打印，否则打印休止符；
- 4) 模拟弹奏：如果键盘有输入且输入为 Ctrl+C，则退出钢琴状态，否则将输入的字符根据数据段的映射转换为相应的音符显示在屏幕上，同时根据用户输入的钢琴按键，调用“sound”子程序播放相应的音符，并将用户的钢琴弹奏记录写入历史记录；
- 5) 结束弹奏：在完成记录后，关闭已经打开的文件，跳转回主菜单循环处，等待用户的下一次操作。

```
;创建文件
lea dx,filename
mov cx,0
mov ah,3ch
int 21h          ;21号中断的3c号功能，用指定的文件名创建一个新文件
mov handle,ax    ;保存文件句柄
mov di,0
input:
; 设置时钟中断频率为每秒2次
mov ah, 00H
mov al, 0Ah
int 15H
print_loop:
; 检测键盘输入
mov ah, 01h
int 16h
jnz begin_get    ; 如果有键盘输入，则跳过打印
; 输出字符
mov ah, 02H
mov dl, '-'
int 21h
; 延迟约0.5秒
mov ah, 86H      ; 设置延迟
mov cx, 5        ; 设置延迟时间，每次循环延迟约0.5秒 (18.2Hz * 0.5 = 9)
int 15H
skip_print:
jmp print_loop   ; 继续循环直到键盘输入检测到
begin_get:
mov ah,00h
int 16h
cmp al,3h        ;按CTRL-C则退出钢琴状态,ah存储键盘扫描码
je finish
;键入转换
mov res,al
```

Fig4. “Play” 子程序

B. “History” 子程序

- 1) 显示提示信息：使用 show mess2 宏显示提示信息，告知用户当前正在播放之前弹奏的音乐；

- 2) 获取记录：调用中断 21h 的 3dh 功能，以只读模式打开历史记录文件，并将文件句柄保存，使用中断 21h 的 3fh 功能，将文件内容读入到数据段定义的 **content** 缓冲区中，最后使用中断 21h 的 3eh 功能，关闭已经打开的文件。
- 3) 播放历史记录：初始化 DI 寄存器为 0，用于遍历 **content** 缓冲区。循环读取 **content** 缓冲区中的字符并将字符转换为 BCD 码，计算该字符对应的音符频率在 **music_table** 表中的地址，然后调用“sound”子程序，DI 寄存器每次加 2，以处理下一个字符。CX 寄存器递减，以控制次数。读起到缓冲区的结尾时，跳转到 **temp** 标签结束播放，执行 **ret** 指令返回主程序。

```
;打开文件
mov ah,3dh
mov al,00h           ;只读
lea dx,filename
mov handle,ax        ;保存文件句柄
int 21h
;读取文件内容
mov ah,3fh
mov bx,handle
lea dx,content
mov cx,100
int 21h
;关闭文件
mov bx,handle
mov ah,3eh
int 21h
;播放
mov di,0
op:
mov al,content[di]
cmp al,0
je temp
sub al,30h           ;将输入的ASCII码转为BCD码
mov bl,al
mov bh,0
add bx,bx            ;数字乘2才是直接定址表中的标号的地址
mov si,music_table[bx]
call sound
```

Fig5. “History” 子程序

C. “sound” 子程序

- 1) 保存寄存器状态：使用 **push** 指令保存 **AX**、**DX** 和 **CX** 寄存器的当前值，以免影响其他部分的程序；
- 2) 播放准备：将 **AL** 加上 30h（将 BCD 码转换为 ASCII 码），将转换后的字符写入 **content** 缓冲区的 **DI** 位置。然后设置 8253 计时器芯片：将 0b6h 写入 8253 控制端口 43h，初始化计时器。将 34dch 存入 **AX** 寄存器，用来计算分频值。使用 **SI** 中的频率值除以 **AX**，结果保存在 **AX** 中，低 8 位和高 8 位分别送入 8253 通道 2 的端口 42h；

- 3) 播放：控制 8255 芯片，从端口 61h 读取当前值到 AL，并保存到 AH。将 AL 的低两位置 1，以打开扬声器开关。将修改后的值写回端口 61h，使扬声器开始发声。同时外部循环 DX 次，每次内部循环 CX 次以创建延迟，使声音持续一段时间；
- 4) 恢复：从 AH 读取原始值并写回端口 61h，关闭扬声器。使用 pop 指令恢复 AX、DX 和 CX 寄存器的原始值。使用 ret 指令返回到调用子程序的地方。

```
add al,30h
mov content[di],al ;写入内存
;8253 芯片(定时/计数器)的设置
mov al,0b6h ;8253初始化
out 43h,al ;43H是8253芯片控制口的端口地址
mov dx,12h
mov ax,34dch
div si ;计算分频值,赋给ax。[si]中存放声音的频率值。
out 42h, al ;先送低8位到计数器, 42h是8253芯片通道2的端口地址
mov al, ah
out 42h, al ;后送高8位计数器
;设置8255芯片, 控制扬声器的开/关
in al,61h ;读取8255 B端口原值
mov ah,al ;保存原值
or al,3 ;使低两位置1, 以便打开开关
out 61h,al ;开扬声器, 发声
mov dx,12 ;保持di时长

wait1:
mov cx, 28000
delay:
nop
loop delay
dec dx
jnz wait1
mov al, ah ;恢复扬声器端口原值
out 61h, al
```

Fig6. “sound” 子程序

D. “music” 子程序

- 1) 播放准备：使用 push 指令保存 AX、BX 、CX 、DX 、SI 和 DI 寄存器的当前值，以免影响其他部分的程序，同时将 freq 和 time 的地址分别加载到 SI 和 DI 寄存器中，准备遍历音符频率和持续时间的数组；
- 2) 检测键盘输入：使用 BIOS 中断 int 16h 检测键盘输入。如果按下空格键，则暂停播放；如果再次按下空格键，则恢复播放。如果按下 CTRL+C 键，则跳转到 end_mus 标签，结束播放；
- 3) 播放音符：通过调用 “sound2” 子程序播放当前频率的音符。使用 int 21h 中断调用，将音符写入到屏幕上，显示当前音符，同时增加 SI 和 DI 寄存器的值，以便读取下一个音符的频率和持续时间，利用循环实现对音乐数组的遍历，直到遇到最后一个音符（频率为 -1）。

- 4) 恢复：使用 `pop` 指令恢复寄存器的原始值，使用 `ret` 指令返回到调用子程序的地方。

```
mov ah,00h
int 16h
cmp al,3h ;按CTRL-C则退出钢琴状态,ah存储键盘扫描码
je end_mus
cmp al, ' '
je pause
jmp ply

pause:
mov ah,00h
int 16h
cmp al, ' '
je ply
jmp pause

ply:
mov dx, [si]
cmp dx, -1;最后一个音符为-1, 结束
je end_mus
call sound2

push si
push bx
push ax
push dx

mov bx, offset freq ; freq_table的起始地址
sub si, bx ; 计算当前频率值在表中的偏移量
shr si, 1 ; 将偏移量除以2, 因为每个频率值占用2个字节
mov dl, note1[si]
mov ah, 02h
int 21h
mov dl, note2[si]
mov ah, 02h
```

Fig7. “music” 子程序

E. “sound2” 子程序

- 1) 保存寄存器状态：使用 `push` 指令保存 `AX`、`DX` 和 `CX` 寄存器的当前值，以免影响其他部分的程序；
- 2) 播放准备：将 `AL` 加上 `30h`（将 `BCD` 码转换为 `ASCII` 码），将转换后的字符写入 `content` 缓冲区的 `DI` 位置。然后设置 `8253` 计时器芯片：将 `0b6h` 写入 `8253` 控制端口 `43h`，初始化计时器。将 `34dch` 存入 `AX` 寄存器，用来计算分频值。使用 `SI` 中的频率值除以 `AX`，结果保存在 `AX` 中，低 8 位和高 8 位分别送入 `8253` 通道 2 的端口 `42h`；
- 3) 播放：控制 `8255` 芯片，从端口 `61h` 读取当前值到 `AL`，并保存到 `AH`。将 `AL` 的低两位置 1，以打开扬声器开关。将修改后的值写回端口 `61h`，使扬声器开始发声。同时外部循环 `DX` 次，每次内部循环 `CX` 次以创建延

迟，使声音持续一段时间，为控制每两个音之间的延迟，将 DI 寄存器的值加载到 DX 中，使用循环和 dec dx 指令实现延迟，直到 DX 减到零；

- 4) 恢复：从 AH 读取原始值并写回端口 61h，关闭扬声器。使用 pop 指令恢复 AX、DX 和 CX 寄存器的原始值。使用 ret 指令返回到调用子程序的地方。

```
mov dx,12h
mov ax,348ch
div word ptr [si] ;计算分频值,赋给ax。[si]中存放声音的频率值
out 42h, al ;先送低8位到计数器
mov al, ah
out 42h, al ;后送高8位计数器

;设置8255芯片,控制扬声器的开/关
in al,61h ;读取8255 B端口原值
mov ah,al ;保存原值到ah
or al,3 ;使低两位置1,以便打开开关
out 61h,al ;开扬声器,发声

mov dx, [di] ;保持[di]时长
wait2:
mov cx, 50000
delay1:
loop delay1
dec dx
jnz wait2

mov al, ah ;恢复扬声器端口原值
out 61h, al
```

Fig8. “Tutory” 子程序

F. “Tutory” 子程序

- 1) 显示信息：使用 show mess3 指令显示钢琴键对应的字符和说明。设置 DL 寄存器为回车符（0Dh）和换行符（0Ah），调用 21h 中断的 2 号功能分别输出回车和换行，实现双重换行；
- 2) 调用 music 子程序：调用 music 子程序播放预设音乐，子程序会依次播放预先设定的频率和时长列表中的音符；
- 3) 调用 play 子程序：调用 play 子程序允许用户输入钢琴按键，子程序将用户按键的字符转换为音符并进行播放；
- 4) 返回菜单：使用 jmp disp 返回到显示菜单的部分，使得系统可以再次接收用户输入的操作选择。

```
show mess3  
;换行  
mov dl,0Dh  
mov ah,2  
int 21h  
mov dl,0Ah  
mov ah,2  
int 21h  
call music  
call play
```

Fig9. “sound2” 子程序

G. “Quit” 子程序

- 1) 退出系统：设置 AH 寄存器为 4Ch，即 DOS 21h 中断的功能号，用于终止程序。调用 21h 中断，使用功能号 4Ch 退出程序，并将控制权返回给操作系统。

```
Quit proc near  
    mov ah,4ch  
    int 21h  
Quit endp
```

Fig10. “Quit” 子程序

源代码清单

1. 子程序功能及参数列表

通过子程序的设计和相互调用，程序能够实现用户输入钢琴按键并记录历史、播放历史记录、展示钢琴键说明并播放预设音乐，以及终止程序的功能。下面是每个子程序的名称，功能，入口参数和出口参数。

子程序	功能	入口参数	出口参数
Play	处理用户输入钢琴按键并记录输入	无	无
History	播放之前记录的音乐	无	无
Tutory	显示钢琴键说明、播放预设音乐	无	无
Quit	终止程序并返回操作系统	无	无
sound	产生音调	si - 音调频率的地址	无
sound2	带有音长的播放音乐	si - 音调频率的地址	无
music	播放预置音乐	无	无

Fig11. 子程序清单

2. 调用关系

A. 主程序

- 1) 程序的入口和主要逻辑流程；
- 2) 负责初始化数据段、调用各个子程序以及处理用户输入。

B. 子程序

- 1) Play: 处理用户钢琴按键输入并记录，调用 sound 播放相应的音符；
- 2) History: 播放之前记录的音乐，调用系统提供的的文件操作；
- 3) Tutory: 显示钢琴键说明，播放预设音乐，并允许用户输入钢琴按键，调用 sound2 和 music。

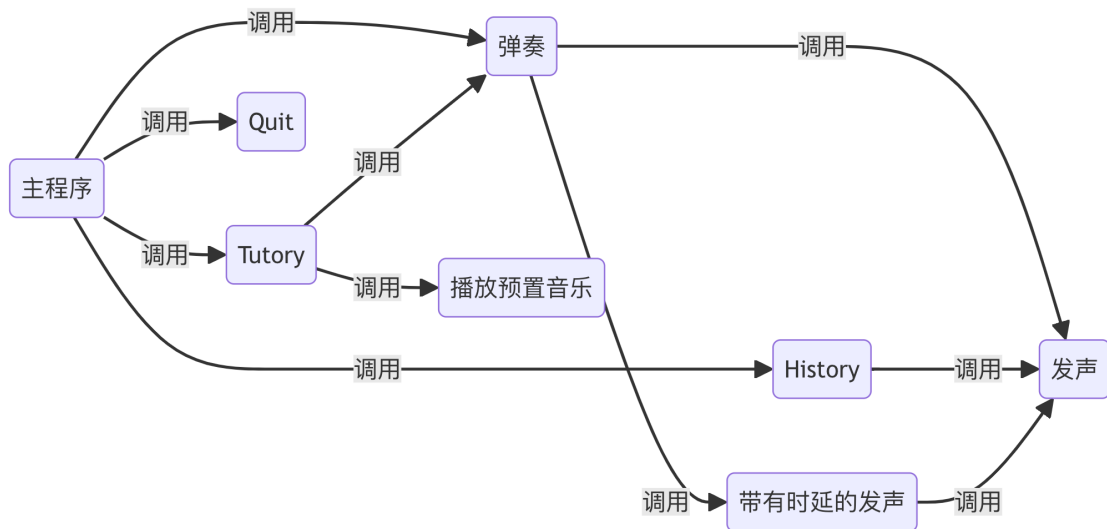


Fig12. 主程序-子程序调用关系

C. 系统调用

- 1) 使用 DOS 中断 21h 实现文件操作和显示字符串等功能；
- 2) 使用 BIOS 中断 15h 实现定时器中断，用于音乐节奏控制；
- 3) 使用键盘中断 16h 实现键盘输入检测。

系统调用	功能
INT 21h/09h	显示字符串
INT 21h/01h	从键盘读取字符
INT 21h/3ch	创建新文件
INT 21h/40h	向文件中写入文本
INT 21h/3dh	打开文件
INT 21h/3fh	读取文件内容
INT 21h/3eh	关闭文件
INT 10h/15h	设置时钟中断频率
INT 10h/02h	输出字符到控制台
INT 10h/86h	设置延迟
INT 16h/01h	检测键盘输入
INT 16h/00h	读取键盘输入
INT 15h	提供延迟功能

Fig13. 系统调用清单

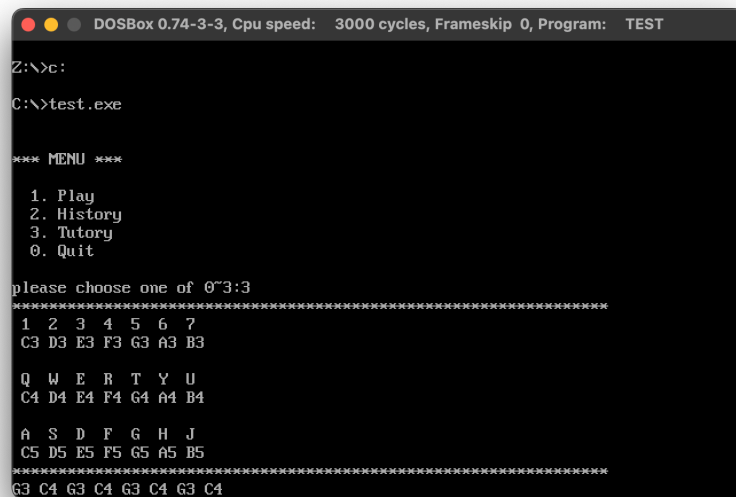
运行结果测试与分析

1. 运行结果测试

下面对每个功能进行测试，由于文档格式限制，暂时无法提供参考音频

A. 新手教程功能

- 1) 预置音乐播放：该功能应该包括展示音符和键盘的对应关系，播放预置音乐以及实时显示播放的音符，其中，按下 CTRL+C 可以退出播放，按下空格可以暂停播放，由下图可见，上述功能均已实现；



```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST
Z:\>c:
C:\>test.exe

*** MENU ***

1. Play
2. History
3. Tutor
0. Quit

please choose one of 0~3:3
=====
1 2 3 4 5 6 7
C3 D3 E3 F3 G3 A3 B3

Q W E R T Y U
C4 D4 E4 F4 G4 A4 B4

A S D F G H J
C5 D5 E5 F5 G5 A5 B5
=====
G3 C4 G3 C4 G3 C4 G3 C4
```

Fig14. 预置音乐播放

- 2) 根据教程进行弹奏：该功能应该包括实时显示播放的音符和每隔 0.5s 打印休止符，由下图可见，上述功能均已实现。

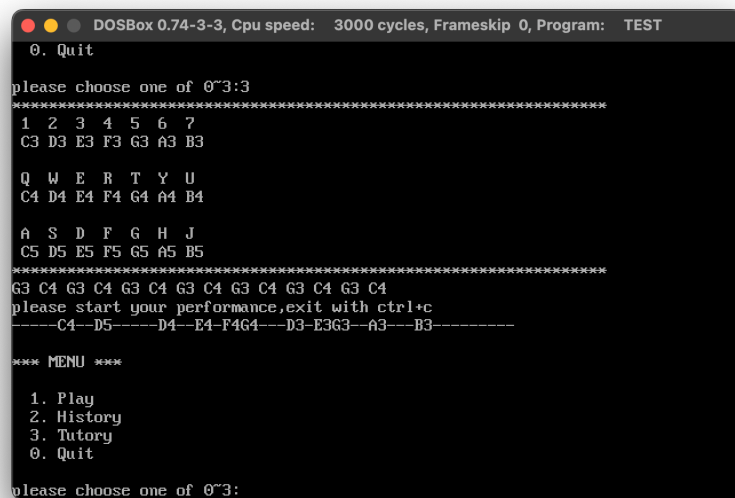


Fig15. 根据教程进行弹奏

B. 弹奏功能和历史记录播放功能

- 1) 弹奏功能：该功能实现与上述“模仿教程进行弹奏”这一功能完全一致，再次不再重复展示；
- 2) 历史记录播放功能：该功能应该包括播放最近一次弹奏的内容和生成一个名为 history 的二进制文件，由下图可见，上述功能均已实现。

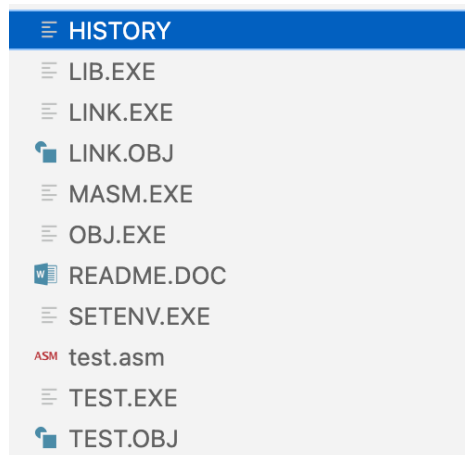


Fig16. 名为 history 的二进制文件



Fig17. 播放最近一次弹奏的内容

C. 退出及容错机制

- 1) 退出：该功能应该包括按下数字 0 即退出系统；
- 2) 容错机制：该功能应该包括按下任意不合法的键，不做任何操作，重新打印菜单并接受输入，由下图可见上述两功能均以实现：

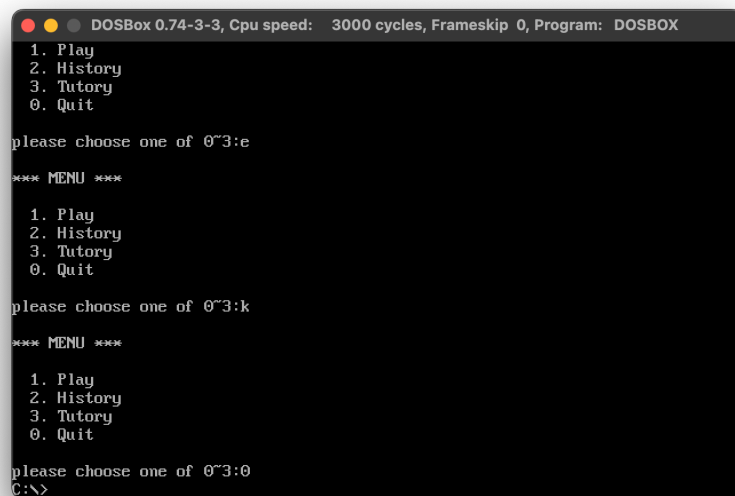


Fig18. 退出及容错机制

结论与心得

1. 程序调试中发现的问题和解决办法

笔者在调试本程序时曾经遇到并解决过以下问题：

A. 键盘输入问题：

- 1) 问题描述：输入空格时播放未能正常停止或开始，按下 CTRL+C 后程序也未正常终止，而是出现乱码；
- 2) 解决方案：经过对寄存器的单步追踪，发现问题出现的原因是寄存器弹栈的顺序与入栈的顺序不匹配，修改顺序后可正常操作。

B. 音符播放不正确：

- 1) 问题描述：弹奏音符后发出声音的频率与设置的频率不一致，个别音符无法发声或发出噪音；
- 2) 解决方案：经过对子程序的检查，发现问题出现的原因是数据段定义与子程序移位逻辑不匹配，导致音符频率和寄存器位置出现错位，修改子程序后可正常操作。

C. 音符播放不正确：

- 1) 问题描述：引入休止符后无法正常播放并显示所弹奏的音符，且无法正常退出弹奏或退出系统；
- 2) 解决方案：经过对休止符打印部分代码的检查，发现问题出现的原因是循环条件设置错误和循环内部逻辑错误，获取到键盘输入后并没有传给发声子程序，且由于寄存器覆盖原因，键盘输入被丢失，导致程序会一直打印休止符，修改循环逻辑后可正常操作。

2. 可能但因时间关系没有来得及完成的想法

笔者在进行本实验时曾经提出过两个想法，由于时间和技术有限，未能完成，但已基本验证其可行性：

A. 历史记录播放优化：

- 1) 问题描述：在加载历史记录时，本系统并未将休止符一并加载，导致播放出的历史记录和用户弹奏时的听感大不相同，十分影响系统体验；
- 2) 可能的解决方案：笔者曾经尝试过使用人耳无法分辨的频率作为休止符的替代加入历史记录的文件中，在依次尝试过超声波和次声波后发现均对原程序有不同程度的干扰，一种可能的替代方案是制定一个不会被弹奏者用到的音符作为休止符，按时序加入历史记录文件中，在读取文件时进行判断，如果遇到此符号则休止 0.5s，但由于时间有限，未做进一步尝试。

B. 音色优化：

- 1) 问题描述：仅由一块 8255 发出的声音并不能满足绝大多数人的要求，需要加入更为丰富的音色层次；
- 2) 可能的解决方案：笔者曾经查询过现代笔记本电脑中发声芯片的个数，无果，大语言模型的回答是大多数电脑上只有一个，但具体情况需要查询相关手册。在有多个芯片以供调用的前提下，笔者构思出两种不同的增色方法，分别是：为同一个音的频率增加少许偏置，利用多个芯片同时发出大同小异的声音；为不同的芯片设置不同的映射逻辑，实现更加复杂的音色层次。由于条件所限，并没有进一步查询和尝试，但确信存在理论上的可能。

3. 实验心得

通过解决本实验中遇到的问题，笔者显著提升了汇编语言的编程和调试技巧，并更为深入的理解了 x86 指令集、寄存器使用和内存操作。此外，笔者现在能熟练使用 DOS 系统调用进行输入输出和文件操作，掌握了通过 I/O 端口进行硬件控制的技巧。笔者也学会了使用汇编语言创建和管理文件，实现数据持久化存储，并通过设置计数器频率和控制扬声器播放不同音符，最后，通过循环和计数器实现程序中的延迟功能，提升时间控制的精确性。这些技能综合起来，将显著增强笔者对汇编语言和系统调用的理解，以及解决其他实际编程问题的能力。