微机系统与接口技术

**姓　　名：** 张金浩

**学　　号：** 201805120629

**专 业：** 计算机科学与技术

**班 级：** 计算机183班

目录

[微机系统与接口技术 1](#_Toc62251679)

[一、程序总体介绍： 3](#_Toc62251680)

[二、大体功能模块： 3](#_Toc62251681)

[三、运用到的所学的知识： 3](#_Toc62251682)

[四、按键说明： 3](#_Toc62251683)

[五、程序流程截图： 4](#_Toc62251684)

[开始页面 4](#_Toc62251685)

[默认状态 6](#_Toc62251686)

[向下移动： 10](#_Toc62251687)

[向左移动： 19](#_Toc62251688)

[向上移动： 19](#_Toc62251689)

[向右移动： 20](#_Toc62251690)

[游戏胜利： 20](#_Toc62251691)

[六、心得体会： 23](#_Toc62251692)

[七、代码： 24](#_Toc62251693)

# 一、程序总体介绍：

程序的主要内容是一个推箱子小游戏，主要的设计思路是结合大二windows程序开发设计的经验，通过汇编程序去实现同样效果，主要利用这学期微机课学到的键盘中断、扬声器中断、绘图模式等相关知识。

# 二、大体功能模块：

1. 欢迎界面模块
2. 像素点绘制模块
3. 各地图元素绘制模块
4. 地图绘制模块
5. 文字打印模块
6. 人物移动模块
7. 墙体检测模块
8. 箱子移动模块
9. 胜利判定模块
10. 胜利音播放模块

# 三、运用到的所学的知识：

1. 字符模式
2. 图像模式
3. 键盘中断
4. 扬声器中断

# 四、按键说明：

|  |  |
| --- | --- |
| 按键 | 说明 |
| W | 向上移动 |
| S | 向下移动 |
| A | 向左移动 |
| D | 向右移动 |
| J | 开始游戏 |

# 五、程序流程截图：

## 开始页面

显示文字和相应用户按键 J 切换启动优秀，主要模块theWelcom

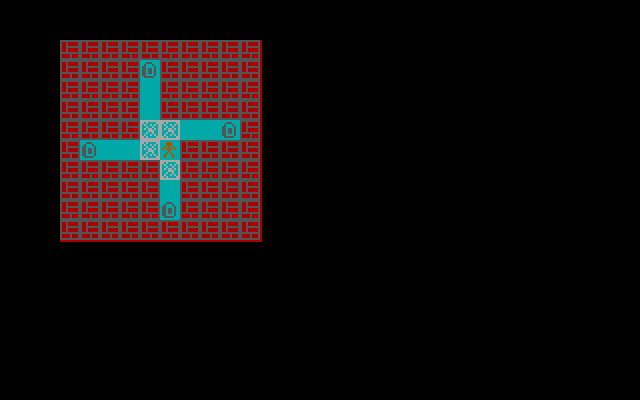


1. ; 欢迎页
2. theWelcom:
3. push ax ; x
4. push bx ; y
5. push dx
6. push si
7. mov ax , 0
8. mov bx , 0
9. mov si , welcofont
10. mov dx , 7
11. mov [ds:willDrawPen] , dx
12. mov dx , 115
13. mov [ds:willDrawRelativeX] , dx
14. mov dx , 70
15. mov [ds:willDrawRelativeY] , dx
17. theWelcomY:
18. cmp bx , [ds:welcofontY]
19. ja theWelcomYEnd
21. theWelcomX:
22. cmp ax , [ds:welcofontX]
23. ja theWelcomXEnd
25. mov dx , 0
26. cmp [si] , dx
27. je drawWelcomPointEnd
29. mov [ds:willDrawX] , ax
30. mov [ds:willDrawY] , bx
32. call drawPointByXAndY
34. drawWelcomPointEnd:
35. inc si
36. inc si
37. inc ax
38. jmp theWelcomX
39. theWelcomXEnd:

42. mov ax , 0
43. inc bx
44. jmp theWelcomY
45. theWelcomYEnd:
47. pop si
48. pop dx
49. pop bx
50. pop ax
51. ret

## 默认状态

默认状态主要应用了绘图函数，为例方便管理，每次重绘我都封装到了一个模块中，叫做drawGameMap



**drawGameMap:**

1. ; 绘制游戏地图
2. drawGameMap:
3. push ax
4. push bx
5. push dx
6. push si
7. push cx
9. ; call clearWind
10. ; call GetIntoDrawMode
11. ; mov ax, 0x0a000
12. ; mov ax, 0xa000
13. ; mov es, ax

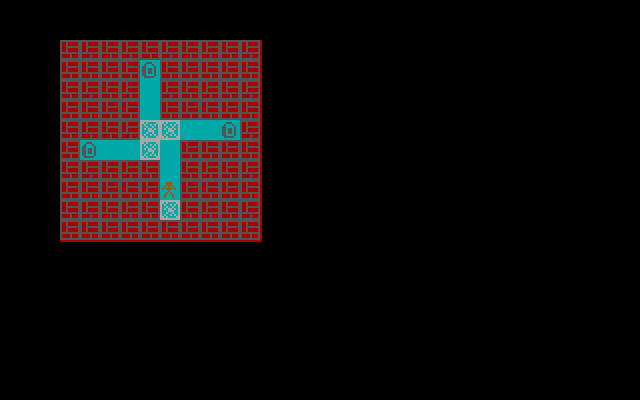
16. mov ax , 0 ; x
17. mov bx , 0 ; y
18. mov si ,  gameMap
19. ; dx 作为中间变量
21. mov dx , [ds:basePointX]
22. mov [ds:nowBasePointX] , dx
24. mov dx , [ds:basePointY]
25. mov [ds:nowBasePointY] , dx

28. drawGameMapY:
29. cmp bx , 9
30. ja drawGameMapYEnd
32. drawGameMapX:
33. cmp ax , 9
34. ja drawGameMapXEnd
35. mov cx , [si]
37. mov dx , [ds:nowBasePointX]
38. add dx , [ds:baseLong]
39. mov [ds:nowBasePointX] , dx
41. mov [ds:willDrawRelativeX] , dx
43. mov dx , [ds:nowBasePointY]
44. mov [ds:willDrawRelativeY] , dx
46. ; 矩阵比较算法
47. cmp cx , 0
48. je drWall
50. cmp cx , 1
51. je drRoad
53. cmp cx , 2
54. je drBox
56. cmp cx , 3
57. je drEndGrain
59. jmp drEd
60. ; 绘图的过程，把偏移量移入，保存
61. ; 调用绘制墙壁
62. drWall:
63. call theWall
64. call theWallGrain
65. jmp drEd
67. ; 调用绘制路面
68. drRoad:
69. call theRoad
70. jmp drEd
72. drBox:
73. call theRoad
74. call theBox
75. jmp drEd
76. drEndGrain:
77. call theRoad
78. call theEndGrain
79. jmp drEd
80. ; 其他调用
81. drEd:
83. drawGamer:
84. judgeX:
85. cmp ax , [ds:baseGamerX]
86. je judgeY
87. jmp drawGamerEnd
88. judgeY:
89. cmp bx , [ds:baseGamerY]
90. je drawGamerStart
91. jmp drawGamerEnd
93. drawGamerStart:
94. call theHum
95. drawGamerEnd:
97. inc ax
98. inc si;
99. inc si;
100. jmp drawGameMapX
101. drawGameMapXEnd:
103. ; x 回到列首部
104. mov dx, [ds:basePointX]
105. mov [ds:nowBasePointX] , dx
107. ; y 去下一个单位
108. mov dx , [ds:nowBasePointY]
109. add dx , [ds:baseLong]
110. mov [ds:nowBasePointY] , dx
112. mov ax , 0
113. inc bx
114. jmp drawGameMapY
115. drawGameMapYEnd:
117. ; 判断胜利
119. mov dx , 1
120. cmp [ds:flagIsWin] , dx
121. jne drawEnd
123. ; 先绘制一个款框
124. ; 先把相对坐标置零
126. mov dx , 150
127. mov [ds:willDrawRelativeX] , dx
128. mov dx , 100
129. mov [ds:willDrawRelativeY] , dx
130. call drawFontGp
132. ; 播放音频以显示胜利
134. call musicFunction

137. ; ; 回到基点位置 ， 根据玩家的位置去绘制
138. drawEnd:
139. pop cx
140. pop si
141. pop dx
142. pop bx
143. pop ax
144. ret

## 向下移动：

几个移动函数的原理一样主要是键盘中断，配合对应的方向值坐标改变，然后调用判断函数和重绘函数实现人物的移动以及墙体检测。主要使用模块，键盘中断int\_key 以及相关的方向对应的方法，还有推箱子的箱子墙体检测judgeNextIsLeagel



**judgeNextIsLeagel：**

1. ; 下一跳的判断
2. ; 流程
3. ; 先判断当前 judgeGamerX and Y 是不是 箱子
4. ; 如果是箱子就判断邻位是否是墙
5. judgeNextIsLeagel:
6. push ax
7. push bx
8. push dx
9. push cx
10. ;stsp 判断是不是箱子
11. mov ax, [ds:judgeGamerX]
12. mov bx, [ds:judgeGamerY]
13. mov [ds:willBoxX] , ax
14. mov [ds:willBoxY] , bx
15. call getGameMapByXAndY
16. ; 获取到下一跳的类型
17. mov cx , [ds:willBoxXAndYBk]
18. mov dx , 2
19. cmp cx , dx
20. jne notIsBox
21. ; 到这里就知道他是一个盒子
22. ; 获取下一跳的情况
23. mov ax , [ds:judgeNextPointX]
24. mov bx , [ds:judgeNextPointY]
25. mov [ds:willBoxX] , ax
26. mov [ds:willBoxY] , bx
27. call getGameMapByXAndY
28. ; 获取到下一跳的类型
29. mov cx , [ds:willBoxXAndYBk]
31. ; 等于零 表示下一跳是 墙，标记为 1 接下来不跳转
32. mov dx , 0
33. cmp cx , 0
34. jne movTheBoxToNext
35. mov dx , 1
36. mov [ds:flagIsBox] , dx
37. jmp notIsBox
38. movTheBoxToNext:
39. ; 这里要把下一跳的矩阵值进行修改
40. ; 具体内容
41. ; judgeNextPointX and Y set 2
42. ; judgeGamerX 1
44. mov ax , [ds:judgeGamerX]
45. mov bx , [ds:judgeGamerY]
46. mov [ds:willBoxX] , ax
47. mov [ds:willBoxY] , bx
48. mov dx , 1
49. mov [ds:newWillBoxXAndYBk] , dx
50. call setGameMapByXAndY
52. mov ax , [ds:judgeNextPointX]
53. mov bx , [ds:judgeNextPointY]
54. mov [ds:willBoxX] , ax
55. mov [ds:willBoxY] , bx
56. mov dx , 2
57. mov [ds:newWillBoxXAndYBk] , dx
58. call setGameMapByXAndY
60. notIsBox:
61. pop cx
62. pop dx
63. pop bx
64. pop ax
65. ret

**int\_key和相应模块检测函数：**

1. ; 键盘 9 好中断
2. int\_key:
3. mov dx, 0x60
4. in  al, dx ; 利用 in  读入输入信息 al 代表 读取的键盘码
6. ; j a4
7. ; 开始游戏
8. cmp al , 0xA4
9. je call\_j
11. ; w 上
12. cmp al , 0x91
13. je call\_w
15. ; s 下
16. cmp al , 0x9F
17. je call\_s
19. ; a 左
20. cmp al , 0x9E
21. je call\_a
23. ; d 右
24. cmp al , 0xA0
25. je call\_d
27. jmp k1
29. ; 响应按键
30. call\_j:
31. call move\_j
32. jmp k1
34. call\_w:
35. call move\_w
36. jmp k1
37. call\_s:
38. call move\_s
39. jmp k1
40. call\_a:
41. call move\_a
42. jmp k1
43. call\_d:
44. call move\_d
45. jmp k1
47. k1:
48. ; 这里触发一下判断
49. ; 这里根据判断进行重绘
50. mov dx, 0x20
51. mov al, 0x61
52. out dx, al ; 写入 cpu
53. iret

56. ; 每次移动需要做的判断
57. ; 接下来的位置是不是墙，是就不可以过去 ok judge wall
58. ; 如果是箱子，判断段箱子有无移动空间 judge move box
59. ; 做一个通关判断，如果数组中没有 3 表示通关 judge is win
61. move\_j:
62. call clearWind
63. call drawGameMap
64. ret
66. ; 按下 弹起 w 上
67. move\_w:
68. push bx
69. push ax
70. push dx
71. ; 逻辑
72. mov ax , [ds:baseGamerX]
73. mov bx , [ds:baseGamerY]
74. dec bx
75. ; 先存到 judge 工具里
76. mov [ds:judgeGamerX] , ax
77. mov [ds:judgeGamerY] , bx

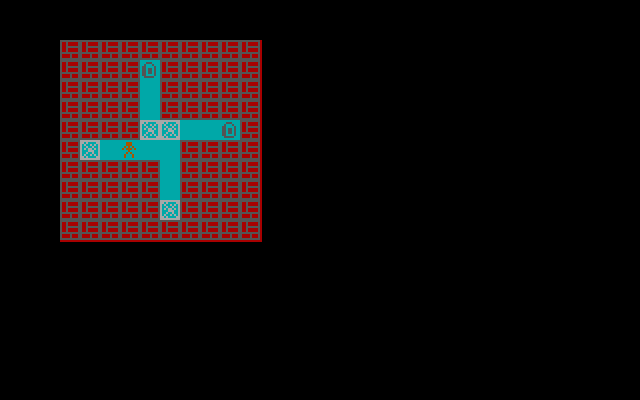
80. call judgeWall
81. ; 判断结束，康康是不是墙，如果是就不做后面的操作
82. mov dx , 1
83. cmp [ds:flagIsWall] , dx
84. je move\_w\_exit

87. ; 保存目标点，以及目标点的对应方向的下一个点
88. ; 用做盒子移动判断
89. mov ax , [ds:judgeGamerX]
90. mov bx , [ds:judgeGamerY]
92. dec bx
94. mov [ds:judgeNextPointX] , ax
95. mov [ds:judgeNextPointY] , bx
97. call judgeNextIsLeagel
99. mov dx , 1
100. cmp [ds:flagIsBox] , dx
101. je move\_w\_exit
103. ; 胜利判断
104. call theWinJudge
106. ; -----------------------
107. mov ax , [ds:judgeGamerX]
108. mov [ds:baseGamerX] , ax
109. mov bx , [ds:judgeGamerY]
110. mov [ds: baseGamerY] , bx
111. call drawGameMap
113. move\_w\_exit:
114. ; 标记初始化
115. mov dx , 0
116. mov [ds:flagIsWall] , dx ; 重置标记
117. mov [ds:flagIsBox] , dx
119. pop dx
120. pop ax
121. pop bx
122. ret
124. ; 按下 弹起 s 下
125. move\_s:
126. push bx
127. push ax
128. push dx
129. mov ax , [ds:baseGamerX]
130. mov bx , [ds:baseGamerY]
131. inc bx
132. mov [ds:judgeGamerX] , ax
133. mov [ds:judgeGamerY] , bx
135. call judgeWall
136. ; 判断结束，康康是不是墙，如果是就不做后面的操作
137. mov dx , 1
138. cmp [ds:flagIsWall] , dx
139. je move\_s\_exit
140. ; call judgeIsBoxS
142. ; 保存目标点，以及目标点的对应方向的下一个点
143. ; 用做盒子移动判断
144. mov ax , [ds:judgeGamerX]
145. mov bx , [ds:judgeGamerY]
147. inc bx
149. mov [ds:judgeNextPointX] , ax
150. mov [ds:judgeNextPointY] , bx
152. call judgeNextIsLeagel
154. mov dx , 1
155. cmp [ds:flagIsBox] , dx
156. je move\_s\_exit
158. ; 胜利判断
159. call theWinJudge
161. ; ----------------------
162. mov ax , [ds:judgeGamerX]
163. mov [ds:baseGamerX] , ax
164. mov bx , [ds:judgeGamerY]
165. mov [ds: baseGamerY] , bx
166. call drawGameMap
168. move\_s\_exit:
169. ; 标记初始化
170. mov dx , 0
171. mov [ds:flagIsWall] , dx ; 重置标记
172. mov [ds:flagIsBox] , dx
174. pop dx
175. pop ax
176. pop bx
177. ret
179. ; 按下 弹起 a 左
180. move\_a:
181. push bx
182. push ax
183. push dx
184. mov ax , [ds:baseGamerY]
185. mov bx , [ds:baseGamerX]
186. dec bx
187. mov [ds:judgeGamerY] , ax
188. mov [ds:judgeGamerX] , bx
190. call judgeWall
191. ; 判断结束，康康是不是墙，如果是就不做后面的操作
192. mov dx , 1
193. cmp [ds:flagIsWall] , dx
194. je move\_a\_exit

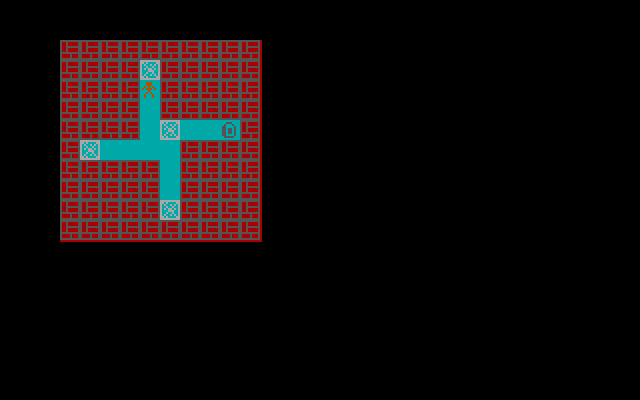
197. ; 保存目标点，以及目标点的对应方向的下一个点
198. ; 用做盒子移动判断
199. mov ax , [ds:judgeGamerX]
200. mov bx , [ds:judgeGamerY]
202. dec ax
204. mov [ds:judgeNextPointX] , ax
205. mov [ds:judgeNextPointY] , bx
207. call judgeNextIsLeagel
209. mov dx , 1
210. cmp [ds:flagIsBox] , dx
211. je move\_a\_exit
213. ; 胜利判断
214. call theWinJudge
216. ; -------------------------
217. mov ax , [ds:judgeGamerX]
218. mov [ds:baseGamerX] , ax
219. mov bx , [ds:judgeGamerY]
220. mov [ds: baseGamerY] , bx
221. call drawGameMap
223. move\_a\_exit:
225. ; 标记初始化
226. mov dx , 0
227. mov [ds:flagIsWall] , dx ; 重置标记
228. mov [ds:flagIsBox] , dx
230. pop dx
231. pop ax
232. pop bx
233. ret

236. ; 按下 d
237. move\_d:
238. push bx
239. push ax
240. push dx
241. mov ax , [ds:baseGamerY]
242. mov bx , [ds:baseGamerX]
243. inc bx
244. mov [ds:judgeGamerY] , ax
245. mov [ds:judgeGamerX] , bx
247. call judgeWall
248. ; 判断结束，康康是不是墙，如果是就不做后面的操作
249. mov dx , 1
250. cmp [ds:flagIsWall] , dx
251. je move\_d\_exit
253. ; 保存目标点，以及目标点的对应方向的下一个点
254. ; 用做盒子移动判断
255. mov ax , [ds:judgeGamerX]
256. mov bx , [ds:judgeGamerY]
258. inc ax
260. mov [ds:judgeNextPointX] , ax
261. mov [ds:judgeNextPointY] , bx
263. call judgeNextIsLeagel
265. mov dx , 1
266. cmp [ds:flagIsBox] , dx
267. je move\_d\_exit
269. ; 胜利判断
270. call theWinJudge
272. ; ---------------------------
273. mov ax , [ds:judgeGamerX]
274. mov [ds:baseGamerX] , ax
275. mov bx , [ds:judgeGamerY]
276. mov [ds: baseGamerY] , bx
277. call drawGameMap
279. move\_d\_exit:
281. ; 标记初始化
282. mov dx , 0
283. mov [ds:flagIsWall] , dx ; 重置标记
284. mov [ds:flagIsBox] , dx
286. pop dx
287. pop ax
288. pop bx
289. ret

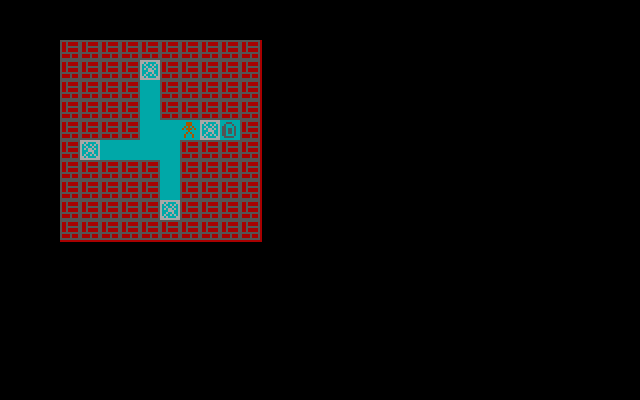
## 向左移动：



## 向上移动：

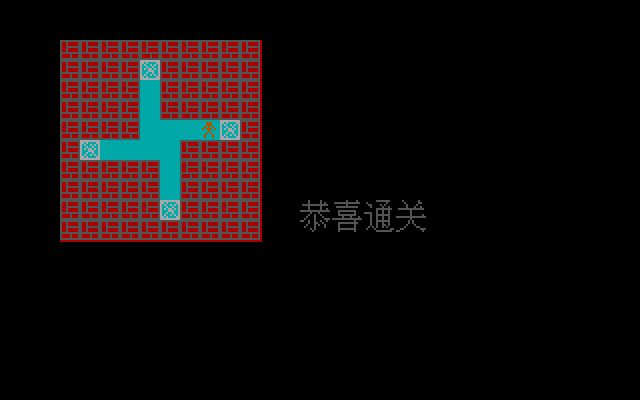


## 向右移动：



## 游戏胜利：

说明：会提示通关并播放一段音频



关键模块：theWinJudge:根据标记为 3 矩阵数据，如果标记为3的数据不存在则表示胜利。调用 musicFunction 模块播放音频和文字书写模块drawFontGp.

**theWinJudge模块：**

1. ; 判断游戏胜利
2. theWinJudge:
3. push ax
4. push bx
5. push dx
6. push si
8. mov dx , 0
9. mov [ds:have3] , dx ; 置零
10. mov ax , 0 ; x
11. mov bx , 0 ; y
12. mov si , gameMap
13. theWinY:
14. cmp bx , 9
15. ja theWinYEnd
16. theWinX:
17. cmp ax , 9
18. ja theWinXEnd
20. mov dx , 3
21. cmp [si] , dx
22. jne theWinXNext
23. mov dx , 1
24. mov [ds:have3] , dx
25. theWinXNext:
26. inc si
27. inc si
28. inc ax
29. jmp theWinX
30. theWinXEnd:
31. mov dx , 0
32. mov ax , 0
33. inc bx
34. jmp theWinY
35. theWinYEnd:
37. mov dx , 1
38. cmp [ds:have3] , dx
39. jne gotWin
40. mov dx , 0
41. mov [ds:flagIsWin] , dx
43. jmp theWinJudgeEnd
44. gotWin:
45. mov dx , 1
46. mov [ds:flagIsWin] , dx
48. theWinJudgeEnd:
49. pop si
50. pop dx
51. pop bx
52. pop ax
53. ret

**musicFunction模块：**

1. ; 播放音频
2. musicFunction:
3. push bx
4. push ax
5. push cx
6. push dx
7. push si
8. ; musizGp
9. mov si , musizGp
10. mov cx , 0
11. musicSt:
12. cmp cx , 10
13. je musicEd
15. mov bx , [si] ; 下面这个 bx 不要用 push pop 屏蔽
16. call tosound
18. inc si
19. inc si
20. inc cx
21. jmp musicSt
22. musicEd:
23. pop si
24. pop dx
25. pop cx
26. pop ax
27. pop bx
28. ret

**drawFontGp:**

1. drawFontGp:
2. push ax
3. push bx
4. push dx
5. push si
6. mov ax , 0
7. mov si , fontdata
8. drawFontStart:
9. cmp ax , [ds:fontSize]
10. ja drawFontEnd
12. mov bx , [si]
13. mov [ds:willDrawX] , bx
14. mov bx , [si+2]
15. mov [ds:willDrawY] , bx
17. call drawPointByXAndY
19. inc si
20. inc si
21. inc si
22. inc si
23. inc ax
24. jmp drawFontStart
25. drawFontEnd:
26. pop si
27. pop dx
28. pop bx
29. pop ax
30. ret

# 六、心得体会：

本次微机课设耗时一天，基本实现了推箱子游戏。因为之前写作业的过程一直追求较好的封装性和通用性，所以写课设的时候基本上根据x和y坐标绘制像素、打印文字、相对坐标绘制等类型的接口都是可以直接拿来使用的，总体的设计思路围绕封装，实现了纯粹的地图绘制以及胜利判断，也为以后程序拓展留下空间。一个学期的微机学习我最深刻的体会就是当我们接触到底层的时候才可以理解设计编写unix、linux、windows等操作系统的前辈的实力，他们的代码和我们的比起来更加精辟，成分利用每一位的数据与内存空间。我认为如果要成为一个合格的开发者，我们就是应该追求良好的代码习惯，不论的代码的命名还是模块的设计都要为后面的开发考虑做到拓展，以及提高代码的可读性。接下来我会继续研究程序开发，去了解操作系统的源码进一步提高个人开发水平。

# 七、代码：

1. org 0x8400
3. ; 进入 13h号 320x200 256色 的 图形模式
4. ;----
5. ;||||
6. ;----
7. ; VM <- 编号 颜色号 0~255
8. ; 字符地址  B8000 H
9. ;           |    | 对应偏移地址
10. ; 0a000 同理
11. jmp start
13. ;-----------------------------------------------------
14. ;---------------------函数变量区-----------------------
15. ;-----------------------------------------------------
16. ; bx 通用寄存器 作为中间变量
17. ;///代码块起始
18. ; TurnXY
19. ; ps : 用到 乘法mul 使用到 ax 寄存器
20. ; 接入 x ， 接入 y , 返回变量
21. TurnX dw 0 ; 接入 x
22. TurnY dw 0 ; 接入 y
23. TurnBK dw 0; 返回 值
25. ; 2.0 版本增加 相对坐标，可以在附加到下面的 实用坐标上
26. willDrawRelativeX dw 0
27. willDrawRelativeY dw 0
29. ; 2020.12.26 1.0 版本
30. willDrawX dw 0
31. willDrawY dw 0
32. willDrawXAndY dw 0
33. willDrawPen db 2
35. ; 声音有关
36. ; 过渡变量
37. nowUseHz dw 600
38. ; hz 转化后的值保存
39. nowUseHzTurned dw 0
40. index db 0 ; 表示当前的标记数
41. dflag db 0 ; 0 小写 ， 1 大写
42. table   dw  262         ;k1 发 **do**    的音，261.6HZ
43. dw  294         ;k2 发 re    的音，293.6HZ
44. dw  330         ;k3 发 mi    的音，329.6HZ
45. dw  349         ;k4 发 fa    的音，349.2HZ
46. dw  392         ;k5 发 sol   的音，392HZ
47. dw  440         ;k6 发 la    的音，440HZ
48. dw  494         ;k7 发 si    的音，493.8HZ
50. musizGp dw 330 , 294 , 262 , 494 , 262 , 392 , 294 , 262 ,  494 , 262
51. ; ------------------------------------------------
52. ;///代码块结束
53. ; ------------------------------------------------

56. ;-----------------------------------------------------
57. ;--------------------通用变量区------------------------
58. ;-----------------------------------------------------
59. ;///代码块起始
60. ;  颜色编号      R     G    B
61. color    db 0,      0 ,   0,   0 ; 背景色 编号 默认 0
62. db 1,      0 ,   0,   0 ; 黑色画笔，清屏用
63. db 2,      49 , 64,  100 ; 线条颜色 编号 2
65. pen db 2 ; 设置画笔颜色
67. points dw 10 , 10 , 11, 10 , 12 ,10 , 13, 10
68. points2 dw 20 , 20 , 10 , 10 , 30 , 30
69. fontSize dw 305
70. ; 点数 205个 0 ~ 204
71. ; 点数 306个 0 ~ 305
72. fontdata dw 4, 0, 10, 0, 23, 0, 28, 0, 38, 0, 39, 0, 40, 0, 41, 0, 42, 0, 43, 0, 44, 0, 51, 0, 59, 0, 4, 1, 10, 1, 12, 1, 17, 1, 18, 1, 19, 1, 20, 1, 21, 1, 22, 1, 23, 1, 24, 1, 25, 1, 26, 1, 27, 1, 28, 1, 29, 1, 33, 1, 43, 1, 52, 1, 59, 1, 60, 1, 1, 2, 2, 2, 3, 2, 4, 2, 5, 2, 6, 2, 7, 2, 8, 2, 9, 2, 10, 2, 11, 2, 12, 2, 13, 2, 23, 2, 34, 2, 35, 2, 40, 2, 42, 2, 53, 2, 58, 2, 4, 3, 10, 3, 18, 3, 19, 3, 20, 3, 21, 3, 22, 3, 23, 3, 24, 3, 25, 3, 26, 3, 27, 3, 28, 3, 35, 3, 41, 3, 44, 3, 53, 3, 57, 3, 60, 3, 4, 4, 10, 4, 13, 4, 38, 4, 39, 4, 40, 4, 41, 4, 42, 4, 43, 4, 44, 4, 45, 4, 49, 4, 50, 4, 51, 4, 52, 4, 53, 4, 54, 4, 55, 4, 56, 4, 57, 4, 58, 4, 59, 4, 60, 4, 61, 4, 0, 5, 1, 5, 2, 5, 3, 5, 4, 5, 5, 5, 6, 5, 7, 5, 8, 5, 9, 5, 10, 5, 11, 5, 12, 5, 13, 5, 14, 5, 19, 5, 20, 5, 21, 5, 22, 5, 23, 5, 24, 5, 25, 5, 26, 5, 27, 5, 38, 5, 41, 5, 44, 5, 55, 5, 5, 6, 9, 6, 19, 6, 27, 6, 32, 6, 33, 6, 34, 6, 35, 6, 38, 6, 41, 6, 44, 6, 55, 6, 4, 7, 7, 7, 10, 7, 19, 7, 20, 7, 21, 7, 22, 7, 23, 7, 24, 7, 25, 7, 26, 7, 27, 7, 35, 7, 38, 7, 39, 7, 40, 7, 41, 7, 42, 7, 43, 7, 44, 7, 55, 7, 61, 7, 3, 8, 7, 8, 11, 8, 21, 8, 25, 8, 29, 8, 35, 8, 38, 8, 41, 8, 44, 8, 48, 8, 49, 8, 50, 8, 51, 8, 52, 8, 53, 8, 54, 8, 55, 8, 56, 8, 57, 8, 58, 8, 59, 8, 60, 8, 61, 8, 62, 8, 2, 9, 7, 9, 12, 9, 13, 9, 14, 9, 16, 9, 17, 9, 18, 9, 19, 9, 20, 9, 21, 9, 22, 9, 23, 9, 24, 9, 25, 9, 26, 9, 27, 9, 28, 9, 29, 9, 30, 9, 35, 9, 38, 9, 41, 9, 44, 9, 55, 9, 0, 10, 1, 10, 4, 10, 7, 10, 10, 10, 13, 10, 27, 10, 35, 10, 38, 10, 39, 10, 40, 10, 41, 10, 42, 10, 43, 10, 44, 10, 54, 10, 56, 10, 4, 11, 7, 11, 8, 11, 11, 11, 19, 11, 20, 11, 21, 11, 22, 11, 23, 11, 24, 11, 25, 11, 26, 11, 27, 11, 28, 11, 35, 11, 38, 11, 41, 11, 44, 11, 54, 11, 56, 11, 3, 12, 7, 12, 9, 12, 12, 12, 19, 12, 27, 12, 35, 12, 38, 12, 41, 12, 42, 12, 44, 12, 53, 12, 57, 12, 2, 13, 7, 13, 12, 13, 19, 13, 27, 13, 34, 13, 36, 13, 38, 13, 41, 13, 43, 13, 52, 13, 58, 13, 59, 13, 5, 14, 7, 14, 19, 14, 20, 14, 21, 14, 22, 14, 23, 14, 24, 14, 25, 14, 26, 14, 27, 14, 33, 14, 37, 14, 45, 14, 46, 14, 50, 14, 51, 14, 60, 14, 61, 14, 62, 14, 6, 15, 19, 15, 27, 15, 38, 15, 39, 15, 40, 15, 41, 15, 42, 15, 43, 15, 44, 15, 45, 15, 48, 15, 49, 15, 61, 15
73. ; 0 墙
74. ; 1 路 普通
75. ; 2 箱子
76. ; 3 终点
77. gameMap dw 0,0,0,0,0,0,0,0,0,0
78. dw 0,0,0,0,3,0,0,0,0,0
79. dw 0,0,0,0,1,0,0,0,0,0
80. dw 0,0,0,0,1,0,0,0,0,0
81. dw 0,0,0,0,2,2,1,1,3,0
82. dw 0,3,1,1,2,1,0,0,0,0
83. dw 0,0,0,0,0,2,0,0,0,0
84. dw 0,0,0,0,0,1,0,0,0,0
85. dw 0,0,0,0,0,3,0,0,0,0
86. dw 0,0,0,0,0,0,0,0,0,0
88. gameEr dw 0,0,0,0,0,0,0,0,0,0
89. dw 0,0,0,1,1,1,0,0,0,0
90. dw 0,0,0,1,1,1,0,0,0,0
91. dw 0,0,1,0,1,0,1,0,0,0
92. dw 0,1,0,1,1,1,0,1,0,0
93. dw 0,0,0,0,1,0,0,0,0,0
94. dw 0,0,0,1,0,1,0,0,0,0
95. dw 0,0,1,0,0,0,1,0,0,0
96. dw 0,0,1,0,0,0,1,0,0,0
97. dw 0,0,0,0,0,0,0,0,0,0
99. boxGp  dw 1,1,1,1,1,1,1,1,1,1
100. dw 1,1,0,1,0,0,0,0,1,1
101. dw 1,0,1,0,0,1,0,1,0,1
102. dw 1,0,0,1,0,0,1,0,0,1
103. dw 1,0,1,0,1,1,0,1,0,1
104. dw 1,0,0,0,1,1,1,0,0,1
105. dw 1,0,0,1,0,0,1,0,0,1
106. dw 1,0,1,0,0,1,0,1,0,1
107. dw 1,1,0,1,0,0,0,0,1,1
108. dw 1,1,1,1,1,1,1,1,1,1
110. endGp  dw 0,0,0,0,0,0,0,0,0,0
111. dw 0,0,0,1,1,1,0,0,0,0
112. dw 0,0,1,0,0,0,1,0,0,0
113. dw 0,1,1,0,0,0,0,1,0,0
114. dw 0,1,1,0,1,1,0,1,0,0
115. dw 0,1,1,0,1,1,0,1,0,0
116. dw 0,1,1,0,1,1,0,1,0,0
117. dw 0,1,1,0,0,0,0,1,0,0
118. dw 0,0,1,1,1,1,1,0,0,0
119. dw 0,0,0,0,0,0,0,0,0,0
121. wallGp dw 1,1,1,1,1,1,1,1,1,1
122. dw 1,0,0,1,0,0,0,0,0,1
123. dw 1,0,0,1,0,0,0,0,0,1
124. dw 1,0,0,1,1,1,1,1,1,1
125. dw 1,0,0,1,0,0,0,0,0,1
126. dw 1,0,0,1,0,0,0,0,0,1
127. dw 1,1,1,1,1,1,1,1,1,1
128. dw 1,0,0,0,0,1,0,0,0,1
129. dw 1,0,0,0,0,1,0,0,0,1
130. dw 1,1,1,1,1,1,1,1,1,1
132. welcofontX dw 89
133. welcofontY dw 29
135. welcofont dw 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
136. dw 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
137. dw 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
138. dw 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
139. dw 0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0
140. dw 0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0
141. dw 0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0
142. dw 0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,1,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0
143. dw 0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,1,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,1,1,1,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0
144. dw 0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0
145. dw 0,0,0,0,0,0,0,0,1,0,1,0,0,1,1,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,1,0,0,0,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0
146. dw 0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,0,1,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
147. dw 0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,1,0,1,1,1,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
148. dw 0,0,0,0,0,0,0,0,1,0,0,1,1,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,1,1,0,1,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0
149. dw 0,0,0,0,0,0,0,0,1,0,1,1,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,0,0,1,1,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0
150. dw 0,0,0,0,0,0,0,0,1,0,1,1,0,1,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0
151. dw 0,0,0,0,0,0,0,0,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
152. dw 0,0,0,0,0,1,1,1,1,1,0,0,0,1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
153. dw 0,0,1,1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,1,1,0,0,0,1,1,1,1,1,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
154. dw 0,0,1,1,1,1,0,0,1,0,0,0,0,1,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,1,1,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
155. dw 0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
156. dw 0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,1,0,1,0,1,1,1,1,1,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
157. dw 0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,0,0,0,1,1,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
158. dw 0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,0,0,0,1,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
159. dw 0,0,0,0,1,1,1,0,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
160. dw 0,0,0,0,0,1,1,1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,1,0,0,0,0,1,1,1,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
161. dw 0,0,0,0,0,0,1,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
162. dw 0,0,0,0,0,0,0,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
163. dw 0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
164. dw 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
166. ; 游戏人物初始位置
167. baseGamerX dw 5
168. baseGamerY dw 5
169. judgeGamerX dw 0
170. judgeGamerY dw 0
172. judgeNextPointX dw 0
173. judgeNextPointY dw 0

176. ; 基点
177. basePointX dw 20
178. basePointY dw 20
179. ; 当前基点
180. nowBasePointX dw 0
181. nowBasePointY dw 0
183. ; 单位长度
184. baseLong dw 10
186. willBoxX dw 0
187. willBoxY dw 0
188. newWillBoxXAndYBk dw 1
189. willBoxXAndYBk dw 1
191. ; 标记取
192. flagIsWall dw 0
193. flagIsBox dw 0
194. flagIsWin dw 0
195. flagSound dw 0
196. have3 dw 0
197. ; bodyWall dw
198. ; ///代码块结束
199. ;-----------------------------------------------------
201. start:
202. ; 进入图形模式
203. call GetIntoDrawMode
204. ; 记录地址
205. mov si , color
206. ; 调用调色模块 draw 一次 遍历 4个地址， color 包含两个 db 所以 运行 两次 可以 修改两组数据
207. call draw
208. call draw
209. call draw
211. ; mov ax, 0x0a000
212. mov ax, 0xa000
213. mov es, ax
214. ; 往显存 (0xa0000 开始)中放入点数据
216. ; call drawGameMap
218. call theWelcom
219. ; ; 键盘中断部分
220. ; mov ax, 0xb800
221. ; mov es,ax
223. mov ax,0
224. mov ds,ax
226. ; 初始化 si
227. mov si, 0
229. cli
230. mov word [ds:0x24] , int\_key
231. mov word [ds:0x26] , 0
232. sti
233. ; 键盘中断部分结束

236. exit:
237. jmp exit
239. ; //////// ; //////// ; //////// ; //////// ; ////////
240. ; //////// ; //////// ; //////// ; //////// ; ////////
241. ; //////// 键盘中断
242. ; //////// ; //////// ; //////// ; //////// ; ////////
243. ; //////// ; //////// ; //////// ; //////// ; ////////
245. ; 键盘 9 好中断
246. int\_key:
247. mov dx, 0x60
248. in  al, dx ; 利用 in  读入输入信息 al 代表 读取的键盘码
250. ; j a4
251. ; 开始游戏
252. cmp al , 0xA4
253. je call\_j
255. ; w 上
256. cmp al , 0x91
257. je call\_w
259. ; s 下
260. cmp al , 0x9F
261. je call\_s
263. ; a 左
264. cmp al , 0x9E
265. je call\_a
267. ; d 右
268. cmp al , 0xA0
269. je call\_d
271. jmp k1
273. ; 响应按键
274. call\_j:
275. call move\_j
276. jmp k1
278. call\_w:
279. call move\_w
280. jmp k1
281. call\_s:
282. call move\_s
283. jmp k1
284. call\_a:
285. call move\_a
286. jmp k1
287. call\_d:
288. call move\_d
289. jmp k1
291. k1:
292. ; 这里触发一下判断
293. ; 这里根据判断进行重绘
294. mov dx, 0x20
295. mov al, 0x61
296. out dx, al ; 写入 cpu
297. iret

300. ; 每次移动需要做的判断
301. ; 接下来的位置是不是墙，是就不可以过去 ok judge wall
302. ; 如果是箱子，判断段箱子有无移动空间 judge move box
303. ; 做一个通关判断，如果数组中没有 3 表示通关 judge is win
305. move\_j:
306. call clearWind
307. call drawGameMap
308. ret
310. ; 按下 弹起 w 上
311. move\_w:
312. push bx
313. push ax
314. push dx
315. ; 逻辑
316. mov ax , [ds:baseGamerX]
317. mov bx , [ds:baseGamerY]
318. dec bx
319. ; 先存到 judge 工具里
320. mov [ds:judgeGamerX] , ax
321. mov [ds:judgeGamerY] , bx

324. call judgeWall
325. ; 判断结束，康康是不是墙，如果是就不做后面的操作
326. mov dx , 1
327. cmp [ds:flagIsWall] , dx
328. je move\_w\_exit

331. ; 保存目标点，以及目标点的对应方向的下一个点
332. ; 用做盒子移动判断
333. mov ax , [ds:judgeGamerX]
334. mov bx , [ds:judgeGamerY]
336. dec bx
338. mov [ds:judgeNextPointX] , ax
339. mov [ds:judgeNextPointY] , bx
341. call judgeNextIsLeagel
343. mov dx , 1
344. cmp [ds:flagIsBox] , dx
345. je move\_w\_exit
347. ; 胜利判断
348. call theWinJudge
350. ; -----------------------
351. mov ax , [ds:judgeGamerX]
352. mov [ds:baseGamerX] , ax
353. mov bx , [ds:judgeGamerY]
354. mov [ds: baseGamerY] , bx
355. call drawGameMap
357. move\_w\_exit:
358. ; 标记初始化
359. mov dx , 0
360. mov [ds:flagIsWall] , dx ; 重置标记
361. mov [ds:flagIsBox] , dx
363. pop dx
364. pop ax
365. pop bx
366. ret
368. ; 按下 弹起 s 下
369. move\_s:
370. push bx
371. push ax
372. push dx
373. mov ax , [ds:baseGamerX]
374. mov bx , [ds:baseGamerY]
375. inc bx
376. mov [ds:judgeGamerX] , ax
377. mov [ds:judgeGamerY] , bx
379. call judgeWall
380. ; 判断结束，康康是不是墙，如果是就不做后面的操作
381. mov dx , 1
382. cmp [ds:flagIsWall] , dx
383. je move\_s\_exit
384. ; call judgeIsBoxS
386. ; 保存目标点，以及目标点的对应方向的下一个点
387. ; 用做盒子移动判断
388. mov ax , [ds:judgeGamerX]
389. mov bx , [ds:judgeGamerY]
391. inc bx
393. mov [ds:judgeNextPointX] , ax
394. mov [ds:judgeNextPointY] , bx
396. call judgeNextIsLeagel
398. mov dx , 1
399. cmp [ds:flagIsBox] , dx
400. je move\_s\_exit
402. ; 胜利判断
403. call theWinJudge
405. ; ----------------------
406. mov ax , [ds:judgeGamerX]
407. mov [ds:baseGamerX] , ax
408. mov bx , [ds:judgeGamerY]
409. mov [ds: baseGamerY] , bx
410. call drawGameMap
412. move\_s\_exit:
413. ; 标记初始化
414. mov dx , 0
415. mov [ds:flagIsWall] , dx ; 重置标记
416. mov [ds:flagIsBox] , dx
418. pop dx
419. pop ax
420. pop bx
421. ret
423. ; 按下 弹起 a 左
424. move\_a:
425. push bx
426. push ax
427. push dx
428. mov ax , [ds:baseGamerY]
429. mov bx , [ds:baseGamerX]
430. dec bx
431. mov [ds:judgeGamerY] , ax
432. mov [ds:judgeGamerX] , bx
434. call judgeWall
435. ; 判断结束，康康是不是墙，如果是就不做后面的操作
436. mov dx , 1
437. cmp [ds:flagIsWall] , dx
438. je move\_a\_exit

441. ; 保存目标点，以及目标点的对应方向的下一个点
442. ; 用做盒子移动判断
443. mov ax , [ds:judgeGamerX]
444. mov bx , [ds:judgeGamerY]
446. dec ax
448. mov [ds:judgeNextPointX] , ax
449. mov [ds:judgeNextPointY] , bx
451. call judgeNextIsLeagel
453. mov dx , 1
454. cmp [ds:flagIsBox] , dx
455. je move\_a\_exit
457. ; 胜利判断
458. call theWinJudge
460. ; -------------------------
461. mov ax , [ds:judgeGamerX]
462. mov [ds:baseGamerX] , ax
463. mov bx , [ds:judgeGamerY]
464. mov [ds: baseGamerY] , bx
465. call drawGameMap
467. move\_a\_exit:
469. ; 标记初始化
470. mov dx , 0
471. mov [ds:flagIsWall] , dx ; 重置标记
472. mov [ds:flagIsBox] , dx
474. pop dx
475. pop ax
476. pop bx
477. ret

480. ; 按下 d
481. move\_d:
482. push bx
483. push ax
484. push dx
485. mov ax , [ds:baseGamerY]
486. mov bx , [ds:baseGamerX]
487. inc bx
488. mov [ds:judgeGamerY] , ax
489. mov [ds:judgeGamerX] , bx
491. call judgeWall
492. ; 判断结束，康康是不是墙，如果是就不做后面的操作
493. mov dx , 1
494. cmp [ds:flagIsWall] , dx
495. je move\_d\_exit
497. ; 保存目标点，以及目标点的对应方向的下一个点
498. ; 用做盒子移动判断
499. mov ax , [ds:judgeGamerX]
500. mov bx , [ds:judgeGamerY]
502. inc ax
504. mov [ds:judgeNextPointX] , ax
505. mov [ds:judgeNextPointY] , bx
507. call judgeNextIsLeagel
509. mov dx , 1
510. cmp [ds:flagIsBox] , dx
511. je move\_d\_exit
513. ; 胜利判断
514. call theWinJudge
516. ; ---------------------------
517. mov ax , [ds:judgeGamerX]
518. mov [ds:baseGamerX] , ax
519. mov bx , [ds:judgeGamerY]
520. mov [ds: baseGamerY] , bx
521. call drawGameMap
523. move\_d\_exit:
525. ; 标记初始化
526. mov dx , 0
527. mov [ds:flagIsWall] , dx ; 重置标记
528. mov [ds:flagIsBox] , dx
530. pop dx
531. pop ax
532. pop bx
533. ret

536. ; 下一跳的判断
537. ; 流程
538. ; 先判断当前 judgeGamerX and Y 是不是 箱子
539. ; 如果是箱子就判断邻位是否是墙
540. judgeNextIsLeagel:
541. push ax
542. push bx
543. push dx
544. push cx
545. ;stsp 判断是不是箱子
546. mov ax, [ds:judgeGamerX]
547. mov bx, [ds:judgeGamerY]
548. mov [ds:willBoxX] , ax
549. mov [ds:willBoxY] , bx
550. call getGameMapByXAndY
551. ; 获取到下一跳的类型
552. mov cx , [ds:willBoxXAndYBk]
553. mov dx , 2
554. cmp cx , dx
555. jne notIsBox
556. ; 到这里就知道他是一个盒子
557. ; 获取下一跳的情况
558. mov ax , [ds:judgeNextPointX]
559. mov bx , [ds:judgeNextPointY]
560. mov [ds:willBoxX] , ax
561. mov [ds:willBoxY] , bx
562. call getGameMapByXAndY
563. ; 获取到下一跳的类型
564. mov cx , [ds:willBoxXAndYBk]
566. ; 等于零 表示下一跳是 墙，标记为 1 接下来不跳转
567. mov dx , 0
568. cmp cx , 0
569. jne movTheBoxToNext
570. mov dx , 1
571. mov [ds:flagIsBox] , dx
572. jmp notIsBox
573. movTheBoxToNext:
574. ; 这里要把下一跳的矩阵值进行修改
575. ; 具体内容
576. ; judgeNextPointX and Y set 2
577. ; judgeGamerX 1
579. mov ax , [ds:judgeGamerX]
580. mov bx , [ds:judgeGamerY]
581. mov [ds:willBoxX] , ax
582. mov [ds:willBoxY] , bx
583. mov dx , 1
584. mov [ds:newWillBoxXAndYBk] , dx
585. call setGameMapByXAndY
587. mov ax , [ds:judgeNextPointX]
588. mov bx , [ds:judgeNextPointY]
589. mov [ds:willBoxX] , ax
590. mov [ds:willBoxY] , bx
591. mov dx , 2
592. mov [ds:newWillBoxXAndYBk] , dx
593. call setGameMapByXAndY
595. notIsBox:
596. pop cx
597. pop dx
598. pop bx
599. pop ax
600. ret
602. ; 判断墙壁判断函数先写这里
603. judgeWall:
604. push ax
605. push bx
606. push si
607. push dx
608. push cx
610. mov si , gameMap
611. mov ax , [ds:judgeGamerY] ; 目标 y
612. mov bx , [ds:judgeGamerX] ; 目标 x
614. mov [ds:willBoxY] , ax
615. mov [ds:willBoxX] , bx
617. call getGameMapByXAndY

620. mov cx , [ds:willBoxXAndYBk] ; 转完之后的结果
622. ; 如果不是墙就移动
623. mov ax , 0
624. cmp cx , ax
625. jne judgeWallNext
627. mov dx , 1
628. mov [ds:flagIsWall] , dx ; 如果不是 0 表示是墙 标记记为 1
629. ; mov bx , [ds:judgeGamerX]
630. ; mov ax , [ds:judgeGamerY] ; 目标 y
632. ; 第二阶段，这个东西不可以直接放进去
633. ; mov [ds:baseGamerX] , bx
634. ; mov [ds:baseGamerY] , ax

637. judgeWallNext:
638. pop cx
639. pop dx
640. pop si
641. pop bx
642. pop ax
643. ret

646. getGameMapByXAndY:
647. push ax
648. push bx
649. push si
650. push cx
651. push dx
652. mov ax , 0 ; y
653. mov bx , 0 ; x
654. mov si , gameMap
655. getGameMapY:
656. cmp ax , 9
657. ja getGameMapYEnd
658. getGameMapX:
659. cmp bx , 9
660. ja getGameMapXEnd
662. judgeBoxsX:
663. cmp bx , [ds:willBoxX]
664. je judgeBoxsY
665. jmp judgeBoxsNext
666. judgeBoxsY:
667. cmp ax , [ds:willBoxY]
668. je saveBoxXAndY
669. jmp judgeBoxsNext
670. saveBoxXAndY:
671. mov dx , [si]
672. mov [ds:willBoxXAndYBk] , dx
673. judgeBoxsNext:
675. inc si
676. inc si
677. inc bx
678. jmp getGameMapX
679. getGameMapXEnd:
681. mov bx , 0
682. inc ax
683. jmp getGameMapY
684. getGameMapYEnd:
686. pop dx
687. pop cx
688. pop si
689. pop bx
690. pop ax
691. ret


695. setGameMapByXAndY:
696. push ax
697. push bx
698. push si
699. push cx
700. push dx
701. mov ax , 0 ; y
702. mov bx , 0 ; x
703. mov si , gameMap
704. setGameMapY:
705. cmp ax , 9
706. ja setGameMapYEnd
707. setGameMapX:
708. cmp bx , 9
709. ja setGameMapXEnd

712. judgeSetBoxsX:
713. cmp bx , [ds:willBoxX]
714. je judgeSetBoxsY
715. jmp judgeSetBoxsNext
716. judgeSetBoxsY:
717. cmp ax , [ds:willBoxY]
718. je saveSetBoxXAndY
719. jmp judgeSetBoxsNext
721. saveSetBoxXAndY:
722. mov dx , [ds:newWillBoxXAndYBk]
723. mov [si] , dx
725. judgeSetBoxsNext:
727. inc si
728. inc si
729. inc bx
730. jmp setGameMapX
731. setGameMapXEnd:
733. mov bx , 0
734. inc ax
735. jmp setGameMapY
736. setGameMapYEnd:
738. pop dx
739. pop cx
740. pop si
741. pop bx
742. pop ax
743. ret
745. ; 判断游戏胜利
746. theWinJudge:
747. push ax
748. push bx
749. push dx
750. push si
752. mov dx , 0
753. mov [ds:have3] , dx ; 置零
754. mov ax , 0 ; x
755. mov bx , 0 ; y
756. mov si , gameMap
757. theWinY:
758. cmp bx , 9
759. ja theWinYEnd
760. theWinX:
761. cmp ax , 9
762. ja theWinXEnd
764. mov dx , 3
765. cmp [si] , dx
766. jne theWinXNext
767. mov dx , 1
768. mov [ds:have3] , dx
769. theWinXNext:
770. inc si
771. inc si
772. inc ax
773. jmp theWinX
774. theWinXEnd:
775. mov dx , 0
776. mov ax , 0
777. inc bx
778. jmp theWinY
779. theWinYEnd:
781. mov dx , 1
782. cmp [ds:have3] , dx
783. jne gotWin
784. mov dx , 0
785. mov [ds:flagIsWin] , dx
787. jmp theWinJudgeEnd
788. gotWin:
789. mov dx , 1
790. mov [ds:flagIsWin] , dx
792. theWinJudgeEnd:
793. pop si
794. pop dx
795. pop bx
796. pop ax
797. ret
798. ; //////// ; //////// ; //////// ; //////// ; ////////
799. ; //////// ; //////// ; //////// ; //////// ; ////////

802. ; //////// ; //////// ; //////// ; //////// ; ////////
803. ; //////// ; //////// ; //////// ; //////// ; ////////
804. ; //////// bit 图标
805. ; //////// ; //////// ; //////// ; //////// ; ////////
806. ; //////// ; //////// ; //////// ; //////// ; ////////
808. ; 绘制游戏地图
809. drawGameMap:
810. push ax
811. push bx
812. push dx
813. push si
814. push cx
816. ; call clearWind
817. ; call GetIntoDrawMode
818. ; mov ax, 0x0a000
819. ; mov ax, 0xa000
820. ; mov es, ax

823. mov ax , 0 ; x
824. mov bx , 0 ; y
825. mov si ,  gameMap
826. ; dx 作为中间变量
828. mov dx , [ds:basePointX]
829. mov [ds:nowBasePointX] , dx
831. mov dx , [ds:basePointY]
832. mov [ds:nowBasePointY] , dx

835. drawGameMapY:
836. cmp bx , 9
837. ja drawGameMapYEnd
839. drawGameMapX:
840. cmp ax , 9
841. ja drawGameMapXEnd
842. mov cx , [si]
844. mov dx , [ds:nowBasePointX]
845. add dx , [ds:baseLong]
846. mov [ds:nowBasePointX] , dx
848. mov [ds:willDrawRelativeX] , dx
850. mov dx , [ds:nowBasePointY]
851. mov [ds:willDrawRelativeY] , dx
853. ; 矩阵比较算法
854. cmp cx , 0
855. je drWall
857. cmp cx , 1
858. je drRoad
860. cmp cx , 2
861. je drBox
863. cmp cx , 3
864. je drEndGrain
866. jmp drEd
867. ; 绘图的过程，把偏移量移入，保存
868. ; 调用绘制墙壁
869. drWall:
870. call theWall
871. call theWallGrain
872. jmp drEd
874. ; 调用绘制路面
875. drRoad:
876. call theRoad
877. jmp drEd
879. drBox:
880. call theRoad
881. call theBox
882. jmp drEd
883. drEndGrain:
884. call theRoad
885. call theEndGrain
886. jmp drEd
887. ; 其他调用
888. drEd:
890. drawGamer:
891. judgeX:
892. cmp ax , [ds:baseGamerX]
893. je judgeY
894. jmp drawGamerEnd
895. judgeY:
896. cmp bx , [ds:baseGamerY]
897. je drawGamerStart
898. jmp drawGamerEnd
900. drawGamerStart:
901. call theHum
902. drawGamerEnd:
904. inc ax
905. inc si;
906. inc si;
907. jmp drawGameMapX
908. drawGameMapXEnd:
910. ; x 回到列首部
911. mov dx, [ds:basePointX]
912. mov [ds:nowBasePointX] , dx
914. ; y 去下一个单位
915. mov dx , [ds:nowBasePointY]
916. add dx , [ds:baseLong]
917. mov [ds:nowBasePointY] , dx
919. mov ax , 0
920. inc bx
921. jmp drawGameMapY
922. drawGameMapYEnd:
924. ; 判断胜利
926. mov dx , 1
927. cmp [ds:flagIsWin] , dx
928. jne drawEnd
930. ; 先绘制一个款框
931. ; 先把相对坐标置零
933. mov dx , 150
934. mov [ds:willDrawRelativeX] , dx
935. mov dx , 100
936. mov [ds:willDrawRelativeY] , dx
937. call drawFontGp
939. ; 播放音频以显示胜利
941. call musicFunction

944. ; ; 回到基点位置 ， 根据玩家的位置去绘制
945. drawEnd:
946. pop cx
947. pop si
948. pop dx
949. pop bx
950. pop ax
951. ret

954. ; 播放音频
955. musicFunction:
956. push bx
957. push ax
958. push cx
959. push dx
960. push si
961. ; musizGp
962. mov si , musizGp
963. mov cx , 0
964. musicSt:
965. cmp cx , 10
966. je musicEd
968. mov bx , [si] ; 下面这个 bx 不要用 push pop 屏蔽
969. call tosound
971. inc si
972. inc si
973. inc cx
974. jmp musicSt
975. musicEd:
976. pop si
977. pop dx
978. pop cx
979. pop ax
980. pop bx
981. ret
983. startMusic:
984. push bx
985. mov bx , [ds:table] ; 下面这个 bx 不要用 push pop 屏蔽
986. call tosound
987. pop bx
988. ret
990. startMusic1:
991. push bx
992. mov bx , [ds:table+4] ; 下面这个 bx 不要用 push pop 屏蔽
993. call tosound
994. pop bx
995. ret
997. drawFontGp:
998. push ax
999. push bx
1000. push dx
1001. push si
1002. mov ax , 0
1003. mov si , fontdata
1004. drawFontStart:
1005. cmp ax , [ds:fontSize]
1006. ja drawFontEnd
1008. mov bx , [si]
1009. mov [ds:willDrawX] , bx
1010. mov bx , [si+2]
1011. mov [ds:willDrawY] , bx
1013. call drawPointByXAndY
1015. inc si
1016. inc si
1017. inc si
1018. inc si
1019. inc ax
1020. jmp drawFontStart
1021. drawFontEnd:
1022. pop si
1023. pop dx
1024. pop bx
1025. pop ax
1026. ret
1028. ; 地面
1029. ; 获取基点 每次循环基点
1030. theRoad:
1031. ; 设置画笔颜色
1032. push ax
1033. push bx
1035. mov ax , 3
1036. mov [ds:willDrawPen] , ax
1038. mov ax , 0 ; x
1039. mov bx , 0 ; y
1040. mov [ds:willDrawX] , ax
1041. mov [ds:willDrawY] , bx
1042. theRoadY:
1043. cmp bx , [ds:baseLong]
1044. ja tehRoadYend
1046. theRoadX:
1047. cmp ax , [ds:baseLong]
1048. ja theRoadXend
1049. mov [ds:willDrawX] , ax
1050. mov [ds:willDrawY] , bx
1051. call drawPointByXAndY
1052. inc ax
1053. jmp theRoadX
1054. theRoadXend:
1056. mov ax , 0
1057. inc bx
1058. jmp theRoadY
1059. tehRoadYend:
1060. pop bx
1061. pop ax
1062. ret
1064. ; 墙
1065. theWall:
1066. ; 设置画笔颜色
1067. push ax
1068. push bx
1070. mov ax , 4
1071. mov [ds:willDrawPen] , ax
1073. mov ax , 0 ; x
1074. mov bx , 0 ; y
1075. mov [ds:willDrawX] , ax
1076. mov [ds:willDrawY] , bx
1077. theWallY:
1078. cmp bx , [ds:baseLong]
1079. ja theWallYend
1081. theWallX:
1082. cmp ax , [ds:baseLong]
1083. ja theWallXend
1084. mov [ds:willDrawX] , ax
1085. mov [ds:willDrawY] , bx
1086. call drawPointByXAndY
1087. inc ax
1088. jmp theWallX
1089. theWallXend:
1091. mov ax , 0
1092. inc bx
1093. jmp theWallY
1094. theWallYend:
1095. pop bx
1096. pop ax
1097. ret
1099. ; 箱子
1100. theBox:
1101. push ax ; x
1102. push bx ; y
1103. push dx
1104. push si
1105. mov ax , 0
1106. mov bx , 0
1107. mov si , boxGp
1108. mov dx , 7
1109. mov [ds:willDrawPen] , dx
1110. theBoxY:
1111. cmp bx , 9
1112. ja theBoxYEnd
1114. theBoxX:
1115. cmp ax , 9
1116. ja theBoxXEnd
1118. mov dx , 0
1119. cmp [si] , dx
1120. je drawBoxPointEnd
1122. mov [ds:willDrawX] , ax
1123. mov [ds:willDrawY] , bx
1125. call drawPointByXAndY
1127. drawBoxPointEnd:
1128. inc si
1129. inc si
1130. inc ax
1131. jmp theBoxX
1132. theBoxXEnd:

1135. mov ax , 0
1136. inc bx
1137. jmp theBoxY
1138. theBoxYEnd:
1140. pop si
1141. pop dx
1142. pop bx
1143. pop ax
1144. ret
1146. ; 人物
1147. theHum:
1148. push ax ; x
1149. push bx ; y
1150. push dx
1151. push si
1152. mov ax , 0
1153. mov bx , 0
1154. mov si , gameEr
1155. mov dx , 6
1156. mov [ds:willDrawPen] , dx
1157. theHumY:
1158. cmp bx , 9
1159. ja theHumYEnd
1161. theHumX:
1162. cmp ax , 9
1163. ja theHumXEnd
1165. mov dx , 0
1166. cmp [si] , dx
1167. je drawHumPointEnd
1169. mov [ds:willDrawX] , ax
1170. mov [ds:willDrawY] , bx
1172. call drawPointByXAndY
1174. drawHumPointEnd:
1175. inc si
1176. inc si
1177. inc ax
1178. jmp theHumX
1179. theHumXEnd:

1182. mov ax , 0
1183. inc bx
1184. jmp theHumY
1185. theHumYEnd:
1187. pop si
1188. pop dx
1189. pop bx
1190. pop ax
1191. ret

1194. ; 终点纹路
1195. theEndGrain:
1196. push ax ; x
1197. push bx ; y
1198. push dx
1199. push si
1200. mov ax , 0
1201. mov bx , 0
1202. mov si , endGp
1203. mov dx , 8
1204. mov [ds:willDrawPen] , dx
1205. theEndGrainY:
1206. cmp bx , 9
1207. ja theEndGrainYEnd
1209. theEndGrainX:
1210. cmp ax , 9
1211. ja theEndGrainXEnd
1213. mov dx , 0
1214. cmp [si] , dx
1215. je drawEndPointEnd
1217. mov [ds:willDrawX] , ax
1218. mov [ds:willDrawY] , bx
1220. call drawPointByXAndY
1222. drawEndPointEnd:
1223. inc si
1224. inc si
1225. inc ax
1226. jmp theEndGrainX
1227. theEndGrainXEnd:

1230. mov ax , 0
1231. inc bx
1232. jmp theEndGrainY
1233. theEndGrainYEnd:
1234. pop si
1235. pop dx
1236. pop bx
1237. pop ax
1238. ret

1241. ; 墙壁纹路
1242. theWallGrain:
1243. push ax ; x
1244. push bx ; y
1245. push dx
1246. push si
1247. mov ax , 0
1248. mov bx , 0
1249. mov si , wallGp
1250. mov dx , 8
1251. mov [ds:willDrawPen] , dx
1252. theWallGrainY:
1253. cmp bx , 9
1254. ja theWallGrainYEnd
1256. theWallGrainX:
1257. cmp ax , 9
1258. ja theWallGrainXEnd
1260. mov dx , 0
1261. cmp [si] , dx
1262. je drawWallPointEnd
1264. mov [ds:willDrawX] , ax
1265. mov [ds:willDrawY] , bx
1267. call drawPointByXAndY
1269. drawWallPointEnd:
1270. inc si
1271. inc si
1272. inc ax
1273. jmp theWallGrainX
1274. theWallGrainXEnd:

1277. mov ax , 0
1278. inc bx
1279. jmp theWallGrainY
1280. theWallGrainYEnd:
1281. pop si
1282. pop dx
1283. pop bx
1284. pop ax
1285. ret
1286. ; //////// ; //////// ; //////// ; //////// ; ////////
1287. ; //////// ; //////// ; //////// ; //////// ; ////////
1289. ; 欢迎页
1290. theWelcom:
1291. push ax ; x
1292. push bx ; y
1293. push dx
1294. push si
1295. mov ax , 0
1296. mov bx , 0
1297. mov si , welcofont
1298. mov dx , 7
1299. mov [ds:willDrawPen] , dx
1300. mov dx , 115
1301. mov [ds:willDrawRelativeX] , dx
1302. mov dx , 70
1303. mov [ds:willDrawRelativeY] , dx
1305. theWelcomY:
1306. cmp bx , [ds:welcofontY]
1307. ja theWelcomYEnd
1309. theWelcomX:
1310. cmp ax , [ds:welcofontX]
1311. ja theWelcomXEnd
1313. mov dx , 0
1314. cmp [si] , dx
1315. je drawWelcomPointEnd
1317. mov [ds:willDrawX] , ax
1318. mov [ds:willDrawY] , bx
1320. call drawPointByXAndY
1322. drawWelcomPointEnd:
1323. inc si
1324. inc si
1325. inc ax
1326. jmp theWelcomX
1327. theWelcomXEnd:

1330. mov ax , 0
1331. inc bx
1332. jmp theWelcomY
1333. theWelcomYEnd:
1335. pop si
1336. pop dx
1337. pop bx
1338. pop ax
1339. ret
1341. ; //////// ; //////// ; //////// ; //////// ; ////////
1342. ; //////// ; //////// ; //////// ; //////// ; ////////
1343. ; //////// 绘图有关接口
1344. ; //////// ; //////// ; //////// ; //////// ; ////////
1345. ; //////// ; //////// ; //////// ; //////// ; ////////
1346. ; 进入绘图模式
1347. GetIntoDrawMode:
1348. ;进入图形模式
1349. mov ah , 00h
1350. mov al , 13h
1351. **int** 10h
1352. ret
1354. ; mov bx, DrawX
1355. ; mov ax, DrawY
1356. ; TurnXY = (y - 0) \* 320 + x
1357. ; x y 联合转化， 将 x y 坐标 通过计算保存起来
1358. ; 2020.12.25 修改，添加屏蔽
1359. TurnXY:
1360. push ax
1361. push bx
1363. mov ax , [ds:TurnY]
1364. mov bx , 320
1365. ; mov bx , 160
1366. mul bx
1367. mov [ds:TurnBK], ax
1368. mov bx , [ds:TurnX]
1369. add [ds:TurnBK],  bx
1370. mov ax , [ds:TurnBK]
1372. pop bx
1373. pop ax
1374. ret


1378. ; 配合 color 进行颜色设置
1379. ; si 数据地址
1380. draw:
1381. push ax
1382. push dx
1384. ; 颜色编号  0x3c8 color 对应
1385. mov dx,0x3c8
1386. mov al , [si]
1387. out dx , al
1388. ; R
1389. mov dx,0x3c9
1390. mov al , [si+1]
1391. out dx , al
1392. ; G
1393. ;mov dx,0x3c9
1394. mov al , [si+2]
1395. out dx , al
1396. ; B
1397. ;mov dx,0x3c9
1398. mov al , [si+3]
1399. out dx , al
1400. add si, 4
1402. pop dx
1403. pop ax
1404. ret

1407. drawPoint:
1408. push bx
1409. mov bx , bp
1410. mov byte[es:bx] , cl ; cl 是颜色编码 这里是绘制 语句
1411. pop bx
1412. ret
1414. clearWind:
1415. push ax
1416. push bx
1417. mov bx,  0
1418. mov ax, 64000
1419. clearStart:
1420. cmp bx , ax
1421. ja clearFinish
1422. mov byte[es:bx] , 1 ; cl 是颜色编码 这里是绘制 语句
1423. inc bx
1424. jmp clearStart
1425. clearFinish:
1426. pop bx
1427. pop ax
1428. ret
1430. ; 3.0 增加相对坐标的绘制方法
1431. ; (willDrawRelativeX , willDrawRelativeY) 为根节点
1432. ; 相对于 这个点进行绘制，上面的点只需要初始化一次
1433. ; 原理 在目标点的基础上增加一层
1434. drawPointByXAndY:
1435. push bp
1436. push bx
1438. mov bx , [ds:willDrawX]
1439. mov bp , [ds:willDrawRelativeX]
1440. add bx , bp
1441. mov [ds:TurnX] , bx
1443. mov bx , [ds:willDrawY]
1444. mov bp , [ds:willDrawRelativeY]
1445. add bx , bp
1446. mov [ds:TurnY] , bx
1448. call TurnXY
1450. mov bx , [ds:TurnBK]
1451. mov [ds:willDrawXAndY] , bx
1453. ; 测试代码
1454. ; mov cl , 2
1455. mov cl , [ds:willDrawPen]
1457. mov bp , [ds:willDrawXAndY]
1458. call drawPoint
1459. pop bx
1460. pop bp
1461. ret
1463. ; //////// ; //////// ; //////// ; //////// ; ////////
1464. ; //////// ; //////// ; //////// ; //////// ; ////////
1465. ; //////// ; //////// ; //////// ; //////// ; ////////

1468. ; //////// ; //////// ; //////// ; //////// ; ////////
1469. ; //////// ; //////// ; //////// ; //////// ; ////////
1470. ; //////// 声音有关的方法
1471. ; //////// ; //////// ; //////// ; //////// ; ////////
1472. ; //////// ; //////// ; //////// ; //////// ; ////////
1474. ; 这里有个大坑，硬件运行太快，所以要用delay（我自己写的扥等待函数）去延迟一下
1475. ; 否则是不会响的
1476. ; 实际的过程是，启动扬声器，关闭发出声音，挺离谱的
1477. tosound:
1478. mov [ds:nowUseHz] , bx
1479. call initSound
1480. call delay
1481. call openSoundPort
1482. call delay
1483. call closeSoundPort
1484. ret
1486. initSound:
1487. mov  al, 10110110B ;初始化，8253计数器2,模式3,初值16位,二进制
1488. out  43H, al ;Q8253的控制口地址43H
1489. call TurnHzToHex
1490. mov  ax, [ds:nowUseHzTurned] ;计数初值=1.19 MHz, 600 Hz=1983
1491. out  42H, al ;送计数初值低位字节，8253的计数器2的地址42H
1492. mov  al, ah
1493. out  42H, al ;送计数初值高位字节
1494. ret

1497. ; 打开扬声器
1498. openSoundPort:
1499. push ax
1500. in al , 61H
1501. or al , 00000011B ; 0000 0011 b
1502. out 61H , al
1503. pop ax
1504. ret
1506. ; todo 写一个关闭扬声器
1507. closeSoundPort:
1508. push ax
1509. in   al,61H
1510. and  al,11111100B   ;并行接口8255，PB1=0,关扬声器
1511. out  61H,al
1512. pop ax
1513. ret

1516. ; 将 hz 转化为可以使用值
1517. ; mubaio
1518. TurnHzToHex:
1519. ; ;设置频率
1520. push ax
1521. push dx
1523. mov dx,12H
1524. mov ax,34DEH
1525. mov di,[ds:nowUseHz]
1526. div di
1527. mov [ds:nowUseHzTurned] , ax
1529. pop dx
1530. pop ax
1531. ret
1533. ; 0.1s 的延迟 用就完事了
1534. delay:
1535. push cx
1536. push ax
1537. push dx
1538. mov  cx, 0FH
1539. mov  dx, 4240H
1540. mov  ah, 86H
1541. **int**  15H
1542. pop dx
1543. pop ax
1544. pop cx
1545. ret
1546. ; //////// ; //////// ; //////// ; //////// ; ////////
1547. ; //////// ; //////// ; //////// ; //////// ; ////////
1548. ; //////// ; //////// ; //////// ; //////// ; ////////