

微 机 设 计 报 告

课程名称： 微型计算机技术及应用

专 业： 计算机科学与技术

班 级： 183班

学 号： 201605010325

姓 名： 林丰俊

指导教师： 舒红波老师

**2021年 1 月23 日**

**目录**

[1.设计目的 1](#_Toc1474)

[2设计流程图： 1](#_Toc3068)

[3.功能模块划分 2](#_Toc12331)

[3.1 初始化： 2](#_Toc21941)

[3.2 主界面输入密码： 2](#_Toc19140)

[3.3数据声明： 5](#_Toc9919)

[3.4 字符流显示： 6](#_Toc23750)

[3.5绘制矩形 8](#_Toc10874)

[3.6矩形移动 9](#_Toc18675)

[3.7 绘制文字 1](#_Toc14131)1

[3.8 清屏 1](#_Toc1944)2

[4截图 1](#_Toc18714)3

[5总结 15](#_Toc19946)

[6源码 1](#_Toc19946)6

# 1.设计目的

网络与计算机领域在生活中扮演着越来越重要的作用。为了让学生更好地了解微型计算机操作系统底层的架构，希望通过本次课程设计能让学生更好的运用汇编语言以及更清楚的了解微机操作系统的底层构建。

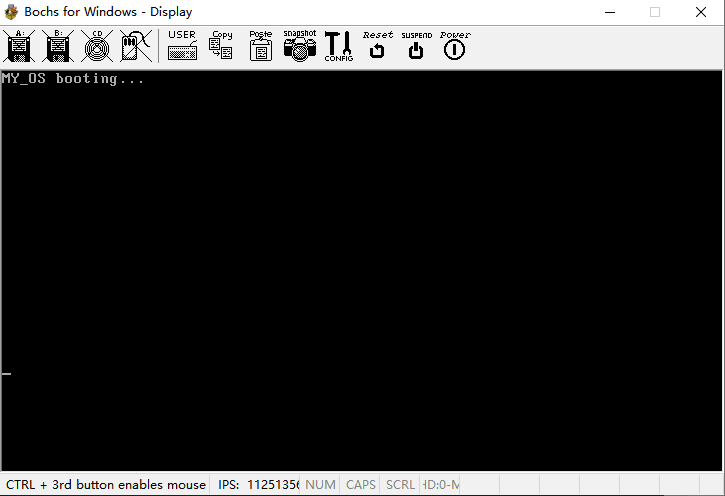
本次设计以课程内容为基础，自我提升水平，自由增加功能，内容包括中断、字符显示、密码显示判断等等。

# 2设计流程图：

# 未命名文件(1)

# 3.功能模块划分

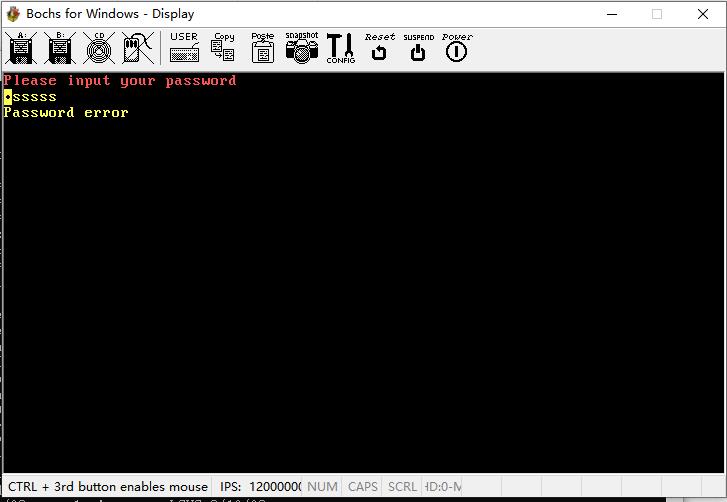
## 3.1 初始化：



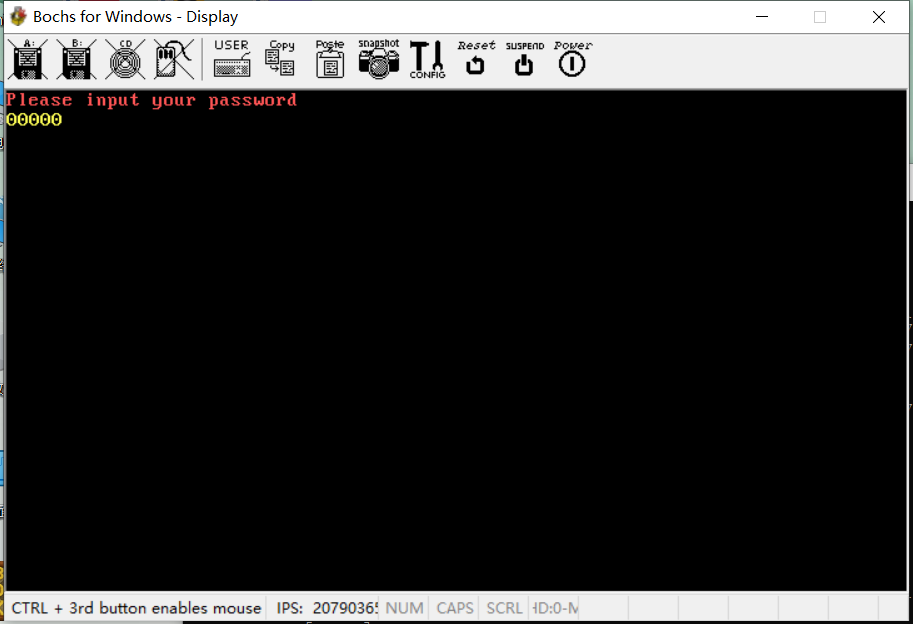
## 3.2 主界面输入密码：

系统进入模块实现密码显示模块从键盘读入字符送AL寄存器。执行时，等待键盘输入，输后入，字符的ASCII码放入AL中。字符与寄存器中存放的字符逐一判断，输入正确则清屏直接进入下一模块，错误则出现错误提示。

（当前展示错误情况）



（当前显示正确情况）



代码如下：

;初始化

\_start:

mov ax, 0b800h ; 显示地址：B8000H

mov es, ax

mov bx, 0 ; 地址

mov ah, 00h ; 背景

clear\_screen:

cmp bx, 4000

JG print\_input\_ready

mov al, ' '

mov [es:bx], ax

inc bx

inc bx

jmp clear\_screen

print\_input\_ready:

mov ax, 0

mov ds, ax

mov di, 0

mov bx, INPUT\_MSG

print\_input:

mov ah, 0ch

cmp di, 50

JG input\_ready ;有符号的大于

mov al, [ds:bx]

mov [es:di], ax

inc di

inc di

inc bx

jmp print\_input

input\_ready:

mov di, 160

mov bx, PASSWD

mov bp, 0

jmp input

input:

cmp di, 170

JG judge\_welcome\_or\_error

mov ah, 0x00

;从键盘读入字符送AL寄存器。执行时，等待键盘输入，一旦输入，字符的ASCII码放入AL中。

;若AL＝0，则AH为输入的扩展码。

int 0x16

mov ah, 0x0e

int 0x10

cmp al, [ds:bx]

JNZ judge\_passwd\_error

mov [es:di], ax

inc di

inc di

inc bx

jmp input

; bp为判断是否输入错误flag

judge\_passwd\_error:

mov bp, 1

mov [es:di], ax

inc di

inc di

inc bx

jmp input

judge\_welcome\_or\_error:

cmp bp, 1

JNZ print\_welcome\_ready

jmp print\_error\_ready

print\_welcome\_ready:

mov bx, WELCOME\_MSG

mov di, 320

jmp print\_welcome

print\_welcome:

cmp di, 334

JG clear\_screen1 ;有符号大于跳转

mov al, [ds:bx]

mov [es:di], ax

inc di

inc di

inc bx

jmp print\_welcome

print\_error\_ready:

mov bx, ERROR\_MSG

mov di, 320

jmp print\_error

print\_error:

cmp di, 346

JG \_end

mov al,[ds:bx]

mov [es:di],ax

inc di

inc di

inc bx

jmp print\_error

\_end:

jmp $

## 3.3数据声明：

提示的数据（INPUT\_MSG、WELCOME\_MSG、ERROR\_MSG），

显示文字的数据（wenzi），

画框的顶点的位置信息（location），

判断的数据（switch）。

代码如下：

;字符串声明

jmp \_start

INPUT\_MSG db 'Please input your password'

WELCOME\_MSG db 'Welcome!'

ERROR\_MSG db 'Password error'

PASSWD db '000000'

;字符矩阵

wenzi:

db 1,0,0,0,0,0,1,1,1,1,1,1,0,1,1,1,1,1,1,1

db 1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0

db 1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0

db 1,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,0,0,0

db 1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0

db 1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0

db 1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0

db 1,1,1,1,1,0,1,0,0,0,0,0,0,1,1,1,1,0,0,0

;矩形位置

location:

dw 16100,16220,54500,54620

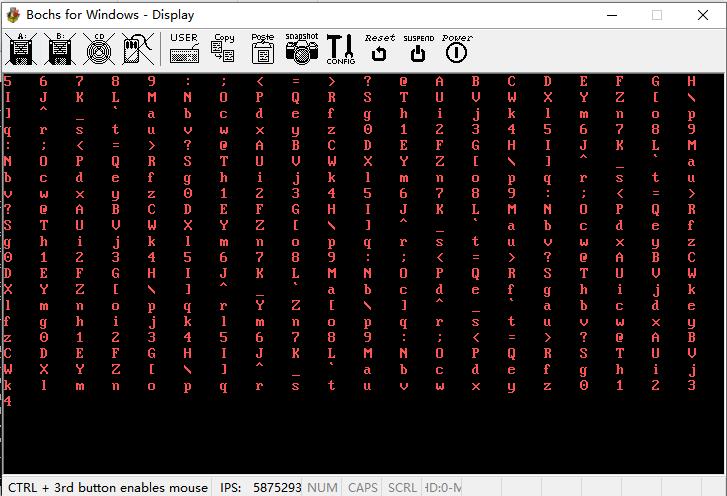
switch:

dw 0

## 3.4 字符流显示：

运用了中断以及把字符输出到显存里，通过按下空格键实现字符流的中断从而进入到另外一个界面。

效果如下：



代码如下：

-------------进入主界面-------------

start0:

sti ;开中断

mov ax, 0

mov si, 0

mov ds, ax ;ds寄存器

mov al, 48 ;像素值

;必须先指定ah寄存器为以下显示服务编号之一

mov ah, 0ch ;以指定需要调用的功用。0CH： 写图形象素

int\_time0:

mov bx, 0b800h ;显存段基址

mov ds, bx ;内存单元的段地址要放在ds中,供后面字符使用

mov [ds:si], ax

inc al，122 ;'a'开始 所有ASCII码字符输出到ASCII 122

add si,8 ;每次位置加8

cmp si, 3200

jbe out1

mov si, 0

out1: cmp al, 122

jbe out2

mov al, 48

out2: push ax

mov al, 0x20 ;横坐标

mov dx, 0x20 ;纵坐标

out dx, al

pop ax

check:

push ax

push bx

push es

call int\_key0

mov ax, cs

mov es, ax

mov bx, switch

mov ax, [es:bx]

cmp ax, 1

je start

pop es

pop bx

pop ax

jmp int\_time0

int\_key0:

push es

push dx

push ax

push bx

mov ah, 01h ;设置光标形状

int 16h ;中断

cmp al, ' '

jne ignore ;不等于' '

mov ax, cs

mov es, ax

mov bx, switch

mov ax, 1 ;显存变为1 画图

mov [es:bx], ax

ignore:

pop bx

pop ax

pop dx

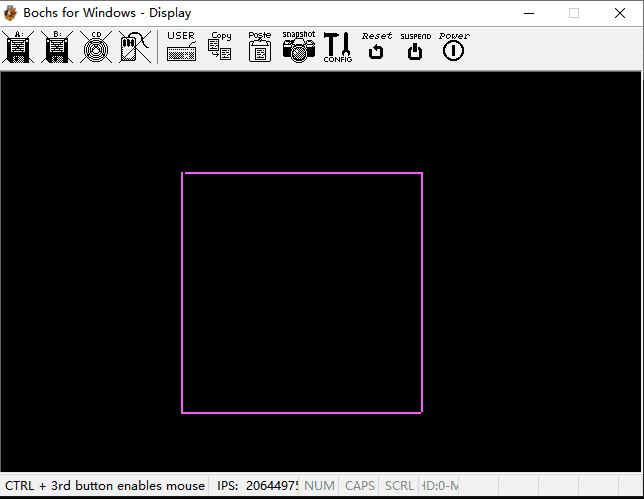
pop es

ret

## 3.5绘制矩形

调用数据声明的location对矩形四个顶点进行初始化，然后构造矩形的四条边，最后通过把显存的0变成1即可实现绘制矩形。

效果如下：



代码如下（绘制画一条线代码）：

push ax

push cx

push di

mov di,location

mov cx,[ds:di]

mov bx, cx

add di,2

mov cx,[ds:di]

mov dx, cx

pop di

pop cx

pop ax

mov ax, cx

add bx, si

add dx, si

loop3:mov [es:bx], ax

inc bx

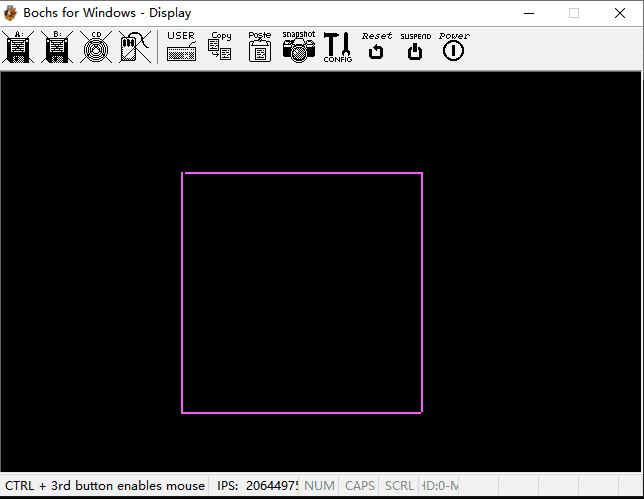
cmp bx, dx

jb loop3

## 3.6矩形移动

键盘的响应，对si变量进行操作，如按‘a’，就实现对si的加操作，最后用add bx, si，add dx, si实现矩形顶点坐标的改变。然后再调用绘制矩形的模块就能实现矩形移动。

效果如下：



代码如下:（以a向左为例）

cmp al, 'a'

jz cha

cha:

sub si,10

jmp paint

paint:

mov al,0x13

mov ah,0x00

int 0x10

mov ax,0a000h

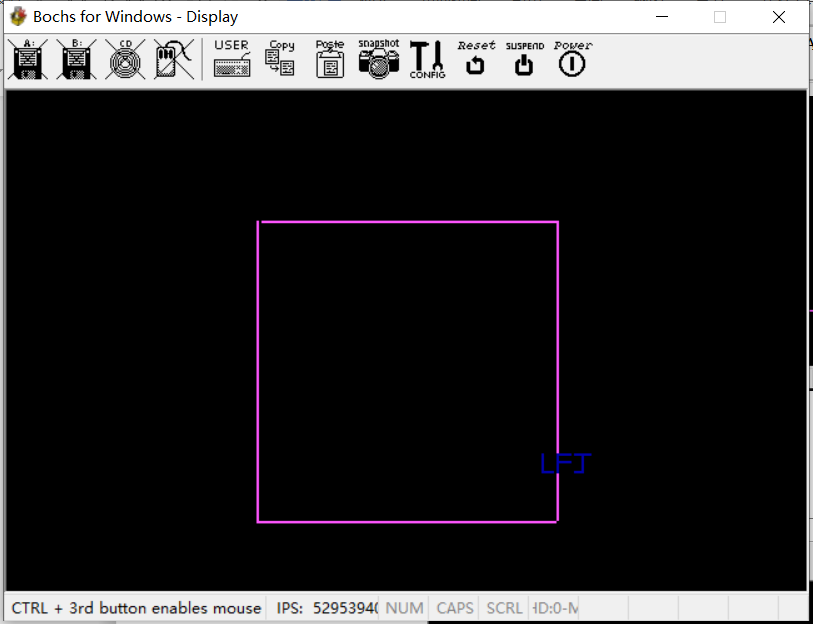
mov es,ax

mov di,0

## 3.7 绘制文字

通过调用数据声明的wenzi，把显存中的值由0变为其他值就可以把文字“写”出来即pig。

效果如下：



代码如下：

；按下k，跳转到hauwenzi

cmp al,'k'

jz huawenzi

；画文字模块

huawenzi:

push cx

mov ax,150

mov bx,350

push ax

push bx

mov ax,cs

mov ds,ax

mov ax,0a000h

mov es,ax

pop bx

pop ax

mov di,ax

mov cx,bx

drawTextMul:

cmp cx,0

je drawTextMain

add di,320

dec cx

jmp drawTextMul

drawTextMain:

push ax

push bx

mov cx,8

mov bx,wenzi

drawTextRow:

push cx

mov cx,20

drawTextCol:

mov al,[ds:bx]

mov [es:di],al

inc di

inc bx

loop drawTextCol

add di,300

pop cx

loop drawTextRow

pop ax

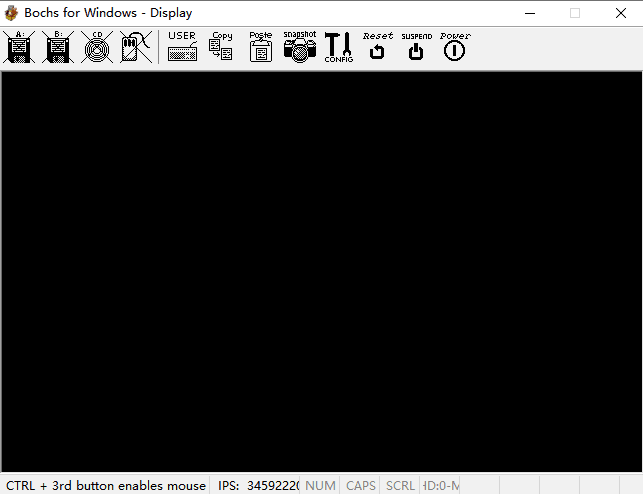
pop bx

pop cx

jmp star

## 3.8 清屏

按下‘c’，就可以清屏，通过把所有的显存中的像素点为0实现效果。



代码如下：

cmp al,'c'

jz clear

clear:

mov di,0

s3:

cmp di, 64000

ja star

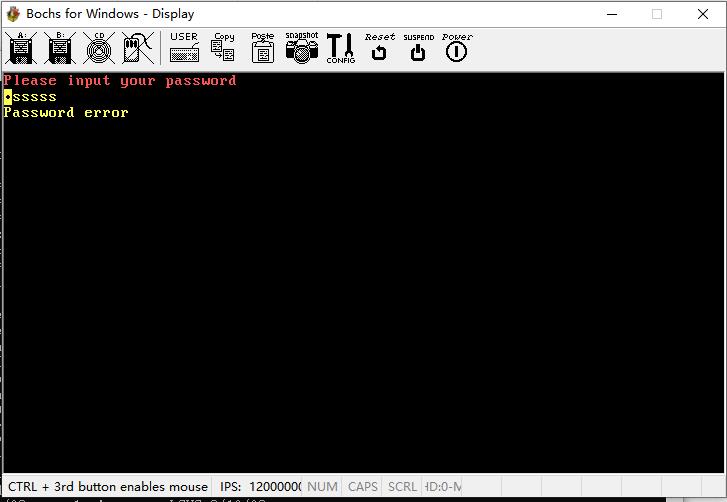
mov al,0

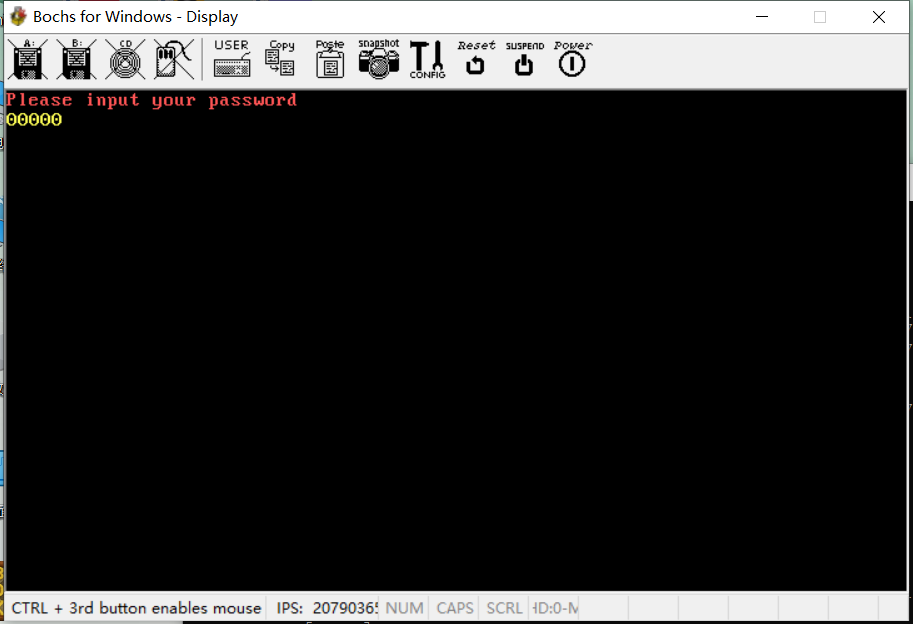
mov [es:di],al

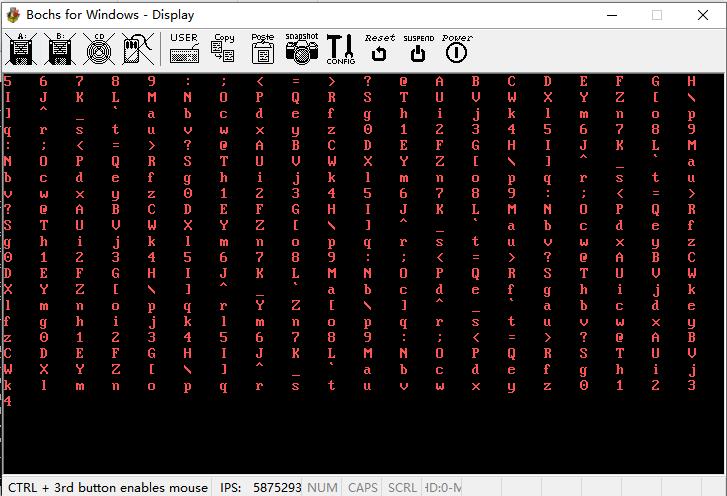
add di,1

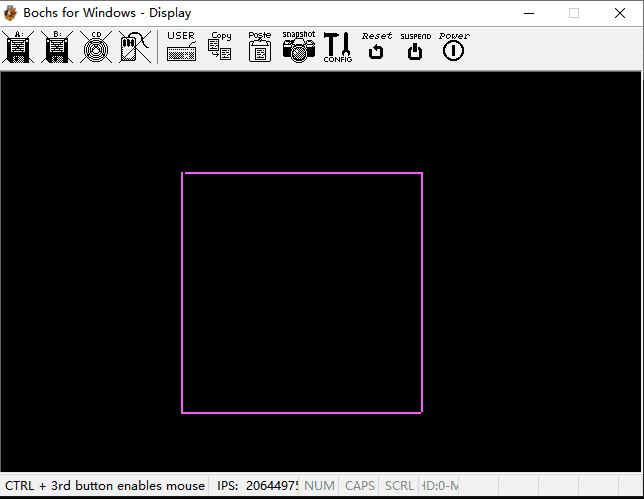
jmp s3

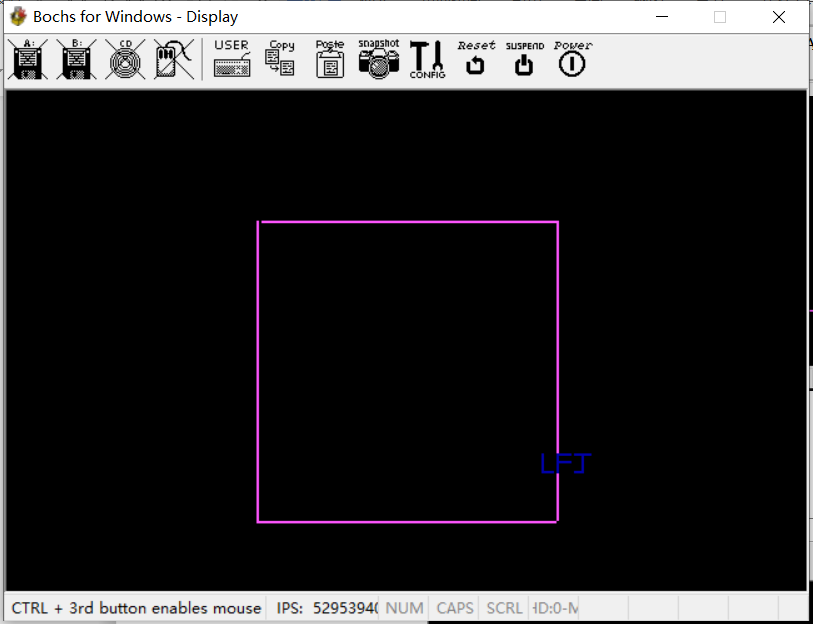
# 4截图

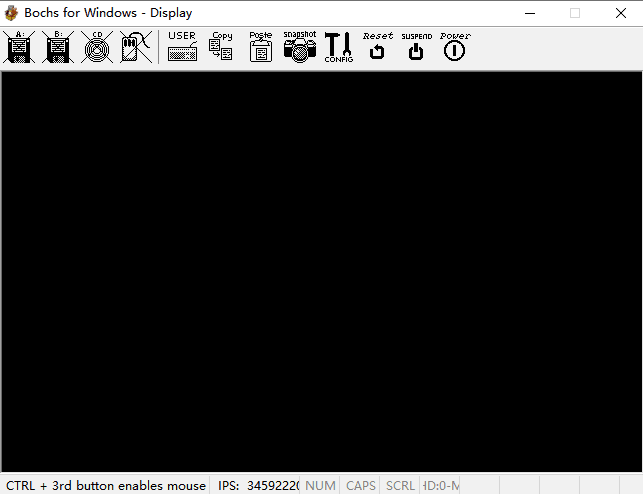












# 5总结

微机的期末课程设计暴露了很多课上没有学好的问题，汇编语言和其他语言还是有很大的不同的。本次课设主要运用的知识有图形的绘制、清屏、字符流中断和文字打印等。

汇编语言的调试过程还是比较折磨的，期间遇到了很多的问题：比如说键盘缓冲区满了，纠结好久发现是刷映像文件的时候没有给键盘留有足够的空间；还有就是密码实现一开始的时候编译通过但是无法输入；再比如说在画矩形的时候，因为push和pop的时候没有一一对应等等。在我的理解汇编这门课程就是合理的运用寄存器，让他为你服务，为你的程序服务，首先你要去理解它，才能更好的掌握它。

这次课程设计和别的同学比较了一下，自己还是有很大的不足的，一个是对于寄存器的原理还是似懂非懂，无法驾驭，另一方面就是对于作品，功能有所欠缺，和其他同学比起来就像一个“畸形儿”。老师上课很认真负责，主要是调动大家的思维去深入理解，但自己平时偷奸耍滑，导致基础不够扎实，在这次的课程设计中自己也没有突破创新的地方，用的都是老师所讲的知识，自己没有深入的去思考、去探索，下一步自己要努力的还很多。

**6源码：**

1. org 0x8400
2. ;字符串声明
3. jmp    \_start
4. INPUT\_MSG     db 'Please input your password'
5. WELCOME\_MSG db 'Welcome!'
6. ERROR\_MSG      db 'Password error'
7. PASSWD         db '000000'
9. ;字符矩阵
10. wenzi:
11. db 1,0,0,0,0,0,1,1,1,1,1,1,0,1,1,1,1,1,1,1
12. db 1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0
13. db 1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0
14. db 1,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,0,0,0
15. db 1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0
16. db 1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0
17. db 1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0
18. db 1,1,1,1,1,0,1,0,0,0,0,0,0,1,1,1,1,0,0,0
20. ;矩形位置
21. location:
22. dw 16100,16220,54500,54620
24. **switch**:
25. dw 0
27. ;-----------------------------
28. ;初始化
29. \_start:
30. mov     ax, 0b800h  ; 显示地址：B8000H
31. mov     es, ax
32. mov     bx, 0        ; 地址
33. mov     ah, 00h     ; 背景
35. clear\_screen:
36. cmp     bx, 4000
37. JG      print\_input\_ready
38. mov     al, ' '
39. mov     [es:bx], ax
40. inc     bx
41. inc     bx
42. jmp     clear\_screen
44. print\_input\_ready:
45. mov     ax, 0
46. mov     ds, ax
47. mov     di, 0
48. mov     bx, INPUT\_MSG
50. print\_input:
51. mov     ah, 0ch
52. cmp     di, 50
53. JG      input\_ready  ;有符号的大于
54. mov     al, [ds:bx]
55. mov     [es:di], ax
56. inc     di
57. inc     di
58. inc     bx
59. jmp     print\_input
61. input\_ready:
62. mov     di, 160
63. mov     bx, PASSWD
64. mov     bp, 0
65. jmp     input
67. input:
68. cmp     di, 170
69. JG      judge\_welcome\_or\_error
70. mov     ah, 0x00            ;从键盘读入字符送AL寄存器。执行时，等待键盘输入，一旦输入，字符的ASCII码放入AL中。
71. ;若AL＝0，则AH为输入的扩展码。
72. **int**     0x16
73. mov     ah, 0x0e
74. **int**     0x10
75. cmp     al, [ds:bx]
76. JNZ     judge\_passwd\_error
77. mov     [es:di], ax
78. inc     di
79. inc     di
80. inc     bx
81. jmp     input
83. ; bp为判断是否输入错误flag
84. judge\_passwd\_error:
85. mov     bp, 1
86. mov     [es:di], ax
87. inc     di
88. inc     di
89. inc     bx
90. jmp     input
92. judge\_welcome\_or\_error:
93. cmp     bp, 1
94. JNZ     print\_welcome\_ready
95. jmp     print\_error\_ready
97. print\_welcome\_ready:
98. mov     bx, WELCOME\_MSG
99. mov     di, 320
100. jmp     print\_welcome
102. print\_welcome:
103. cmp     di, 334
104. JG      clear\_screen1           ;有符号大于跳转
105. mov     al, [ds:bx]
106. mov     [es:di], ax
107. inc     di
108. inc     di
109. inc     bx
110. jmp     print\_welcome
112. print\_error\_ready:
113. mov     bx, ERROR\_MSG
114. mov     di, 320
115. jmp     print\_error
117. print\_error:
118. cmp     di, 346
119. JG      \_end
120. mov     al,[ds:bx]
121. mov     [es:di],ax
122. inc     di
123. inc     di
124. inc     bx
125. jmp     print\_error
127. \_end:
128. jmp $
129. ;-------------------------
131. clear\_screen1:                      ;进入主界面前的清屏
132. mov     ah, 0x00
133. mov     al, 0x03
134. **int**     0x10
136. ;-------------进入主界面-------------
137. start0:
138. sti             ;开中断
139. mov ax, 0
140. mov si, 0
141. mov ds, ax          ;ds寄存器
142. mov al, 48          ;像素值
143. ;必须先指定ah寄存器为以下显示服务编号之一
144. mov ah, 0ch         ;以指定需要调用的功用。0CH： 写图形象素
145. int\_time0:
146. mov bx, 0b800h          ;显存段基址
147. mov ds, bx          ;内存单元的段地址要放在ds中,供后面字符使用
148. mov [ds:si], ax
150. inc al              ;'a'开始 所有ASCII码字符输出 到ASCII  122
151. add si,8                ;每次位置加8
153. cmp si, 3200
154. jbe out1
155. mov si, 0
157. out1:   cmp al, 122
158. jbe out2
159. mov al, 48
161. out2:   push ax
162. mov al, 0x20            ;横坐标
163. mov dx, 0x20            ;纵坐标
164. out dx, al
165. pop ax
167. check:
168. push ax
169. push bx
170. push es
172. call int\_key0
173. mov ax, cs
174. mov es, ax
175. mov bx, **switch**
176. mov ax, [es:bx]
177. cmp ax, 1
178. je start
180. pop es
181. pop bx
182. pop ax
184. jmp int\_time0
186. int\_key0:
187. push es
188. push dx
189. push ax
190. push bx
192. mov ah, 01h         ;设置光标形状
193. **int** 16h                 ;中断
195. cmp al, ' '
196. jne ignore          ;不等于' '
198. mov ax, cs
199. mov es, ax
200. mov bx, **switch**
201. mov ax, 1
202. mov [es:bx], ax
203. ignore:
204. pop bx
205. pop ax
206. pop dx
207. pop es
208. ret

211. start:
212. pop es
213. pop bx
214. pop ax
216. mov cx,13
217. mov si,0
218. mov ah, 00h         ;设置显示器模式
219. mov al, 13h         ;640X480 256色
220. **int** 10h             ;中断
222. mov ax, cs
223. mov ds, ax
225. star:                   ;每次循环接受输入
226. mov ah, 0
227. **int** 16h             ;从键盘接受一个字符
229. cmp al, 'c'
230. jz clear
231. cmp al, 'a'
232. jz cha
233. cmp al, 'd'
234. jz chd
235. cmp al, 'w'
236. jz chw
237. cmp al, 's'
238. jz chs
240. cmp al, 'k'
241. jz huawenzi

244. jmp star

247. chp:
248. mov cx,56
249. jmp paint
251. cho:
252. mov cx,28
253. jmp paint

256. cha:
257. sub si,10
258. jmp paint
260. chd:
261. add si,10
262. jmp paint
264. chw:
265. sub si,640
266. jmp paint
268. chs:
269. add si,640
270. jmp paint
272. paint:
273. mov al,0x13
274. mov ah,0x00
275. **int** 0x10
277. mov ax,0a000h
278. mov es,ax
279. mov di,0
281. x:
282. s4:
283. cmp di, 64000
284. ja rect
285. mov al,0
286. mov [es:di],al
287. add di,1
288. jmp s4
290. rect:
291. push ax
292. push cx
293. push di
295. mov di,location
296. mov cx,[ds:di]
297. mov bx, cx
298. add di,2
299. mov cx,[ds:di]
300. mov dx, cx
302. pop di
303. pop cx
304. pop ax
305. mov ax, cx
306. add bx, si
307. add dx, si
309. loop3:  mov [es:bx], ax
310. inc bx
311. cmp bx, dx
312. jb loop3
314. push ax
315. push cx
316. push di
318. mov di,location
319. mov cx,[ds:di]
320. mov bx, cx
321. add di,4
322. mov cx,[ds:di]
323. mov dx, cx
325. pop di
326. pop cx
327. pop ax

330. ;mov bx, 16100
331. ;mov dx, 54500
332. add bx, si
333. add dx, si
335. loop4:  mov [es:bx], ax
336. add bx, 320
337. cmp bx, dx
338. jb loop4

341. push ax
342. push cx
343. push di
345. mov di,location
346. add di,2
347. mov cx,[ds:di]
348. mov bx, cx
349. add di,4
350. mov cx,[ds:di]
351. mov dx, cx
352. pop di
353. pop cx
354. pop ax
356. ;mov bx, 16220
357. ;mov dx, 54620
358. add bx, si
359. add dx, si
361. loop5:  mov [es:bx], ax
362. add bx, 320
363. cmp bx, dx
364. jb loop5

367. push ax
368. push cx
369. push di
370. mov di,location
371. add di,4
372. mov cx,[ds:di]
373. mov bx, cx
374. add di,2
375. mov cx,[ds:di]
376. mov dx, cx
377. pop di
378. pop cx
379. pop ax
381. ;mov bx, 54500
382. ;mov dx, 54620
383. add bx, si
384. add dx, si
385. loop6:
386. mov [es:bx], ax
387. inc bx
388. cmp bx, dx
389. jb loop6
391. jmp star
393. huawenzi:
394. push cx
395. mov ax,150
396. mov bx,350
398. push ax
399. push bx
401. mov ax,cs
402. mov ds,ax
404. mov ax,0a000h
405. mov es,ax
407. pop bx
408. pop ax
409. mov di,ax
410. mov cx,bx
412. drawTextMul:
413. cmp cx,0
414. je drawTextMain
415. add di,320
416. dec cx
417. jmp drawTextMul
419. drawTextMain:
420. push ax
421. push bx
422. mov cx,8
423. mov bx,wenzi
425. drawTextRow:
426. push cx
427. mov cx,20
429. drawTextCol:
430. mov al,[ds:bx]
431. mov [es:di],al
432. inc di
433. inc bx
434. loop drawTextCol
435. add di,300
436. pop cx
437. loop drawTextRow
439. pop ax
440. pop bx
441. pop cx
442. jmp star
444. clear:
445. mov di,0