



本科生实验报告

实验课程: 操作系统

实验名称: 实验二

专业名称: 计算机科学与技术（超算）

学生姓名: 钟芳婷

学生学号: 19335290

实验地点: D503

报告时间: 2021. 3. 23

一、实验概述

在第一章中，同学们会学习到 x86 汇编、计算机的启动过程、IA-32 处理器架构和字符显存原理。根据所学的知识，同学们能自己编写程序，然后让计算机在启动后加载运行，以此增进同学们对计算机启动过程的理解，为后面编写操作系统加载程序奠定基础。同时，同学们将学习如何使用 gdb 来调试程序的基本方法。

二、实验要求

- 1. 独立完成实验四个任务：MBR、实模式中断、汇编、汇编小程序
- 2. 实验不限语言， C/C++/Rust 都可以。
- 3. 实验不限平台， Windows、Linux 和 MacOS 等都可以。
- 4. 实验不限 CPU， ARM/Intel/Risc-V 都可以。

三、实验原理

(一)IA-32 处理器

IA-32 处理器是指从 Intel 80386 开始到 32 位的奔腾 4 处理器，是最为经典的处理器架构。至此，Intel 32 位的处理器也被称为 x86 处理器。IA-32 处理器其有三种基本操作模式：保护模式、实地址模式(简称实模式)、系统管理模式和虚拟 8086 模式。我们在操作系统实验过程中仅用到实模式和保护模式。

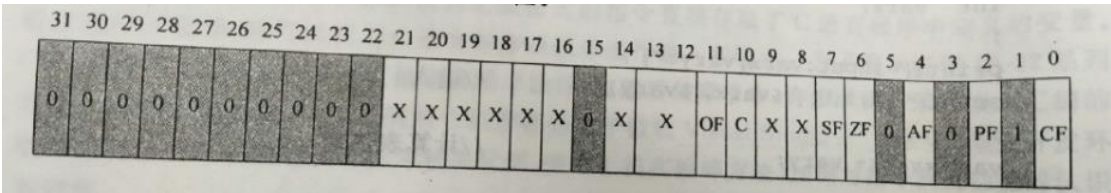
IA-32 处理器的重要组成部分如下。

- **地址空间**。保护模式和实模式最大的不同在于地址总线。实模式使用 20 位地址总线、16 位寄存器；保护模式使用 32 位地址总线、32 位寄存器。因此，实模式的寻址空间为 $2^{20}=1\text{MB}$ ，保护模式的寻址空间为 $2^{32}=4\text{GB}$ 。这里的地址指的是内存地址。
- **基本寄存器**。寄存器是 CPU 内部的高速存储单元。IA-32 处理器主要有 8 个通用寄存器 `eax, ebx, ecx, edx, ebp, esp, esi, edi`、6 个段寄存器 `cs, ss, ds, es, fs, gs`、标志寄存器 `eflags`、指令地址寄存器 `eip`。
- **通用寄存器**。通用寄存器有 8 个，分别是 `eax, ebx, ecx, edx, ebp, esp, esi, edi`，均是 32 位寄存器。通用寄存器用于算术运算和数据传输。32 位寄存器用于保护模式，为了兼容 16 位的实模式，每一个 32 位寄存器又可以拆分成 16 位寄存器和 8 位寄存器来访问。例如 `ax` 是 `eax` 的低 16 位，`ah` 是 `ax` 高 8 位，`al` 是 `ax` 的低 8 位。`ebx, ecx, edx` 也有相同的访问模式。

	31	16	15	8	7	0	
EAX					AH	AL	AX
EBX					BH	BL	BX
ECX					CH	CL	CX
EDX					DH	DL	DX
ESI					SI		
EDI					DI		
EBP					BP		
ESP					SP		

有名称的通用寄存器，
可以独立访问；
否则，不行

- **段寄存器**。段寄存器有 cs,ss,ds,es,fs,gs，用于存放段的基地址，段实际上就是一块连续的内存区域。
- **指令指针**。cip 存放下一条指令的地址。有些机器指令可以改变 cip 的地址，导致程序向新的地址进行转移，如 ret 指令。
- **状态寄存器**。eflags 存放 CPU 的一些状态标志位。下面提到的标志如进位标志实际上是 eflags 的某一个位。常用的标志位如下。



- 进位标志(CF)。在无符号算术运算的结果无法容纳于目的操作数时被置 1。
- 溢出标志(OF)。在有符号算术运算的结果无法容纳于目的操作数时被置 1。
- 符号标志(SF)。在算术或逻辑运算产生的结果为负时被置 1。
- 零标志(ZF)。在算术或逻辑运算产生的结果为 0 时被置 1。

Register	作用	Sub Register
eax	累加，算数逻辑	ax,ah,al
ebx	基址，数组	bx,bh,bl
ecx	计数，迭代	cx,ch,cl
edx	数据，算数	dx,dh,dl
esi	源索引，数组	si
edi	目的索引，数组	di
esp	堆栈，栈顶指针	sp
ebp	堆栈，栈底指针	bp
eip	指令，指向next指令	ip
eflags	标志位，状态控制	flags

(二) 实地址模式

在实地址模式下，IA-32 处理器使用 20 位的地址线，可以访问 $2^{20}=1\text{MB}$ 的内存，范围时 0x0000 到 0xFFFFF。但是，我们看到寄存器的访问模式只有 32 位，16 位和 8 位，形如 `cax`，`ax`，`ah`，`al`。那么我们如何才能使用 16 位的寄存器表示 20 位的地址空间呢？这在当时也给 Intel 工程师带来了极大的困扰，但是聪明的工程师想出来一种“段地址+偏移地址”的解决方案。段地址和偏移地址均为 16 位。此时，一个 1MB 中的地址，称为物理地址，按如下方式计算出来：

$$\text{物理地址}=(\text{段地址} \ll 4)+\text{偏移地址}$$

实模式下，物理地址可以记为“段地址:偏移地址”。段地址和偏移地址均用 16 位表示，最大值均为 0xFFFF。因此，每个段的最大长度是 64KB。按上述计算方式的可表示的最大地址是大于 20 位地址线表示的 1MB 内存空间的，因此满足要求。

段地址存放在段寄存器 cs, ds, es, ss 中，在编程中我们给出的地址(如下面提到的数据标号和代码标号)实际上是偏移地址，当我们要寻址时，CPU 会自动根据偏移地址的类型如栈段、数据段和代码段来从对应的段寄存器中取出段地址，然后和偏移地址一起，计算出物理地址，CPU 最终使用物理地址进行寻址。

(三) 计算机开机启动过程

经典的计算机的启动分为以下步骤。

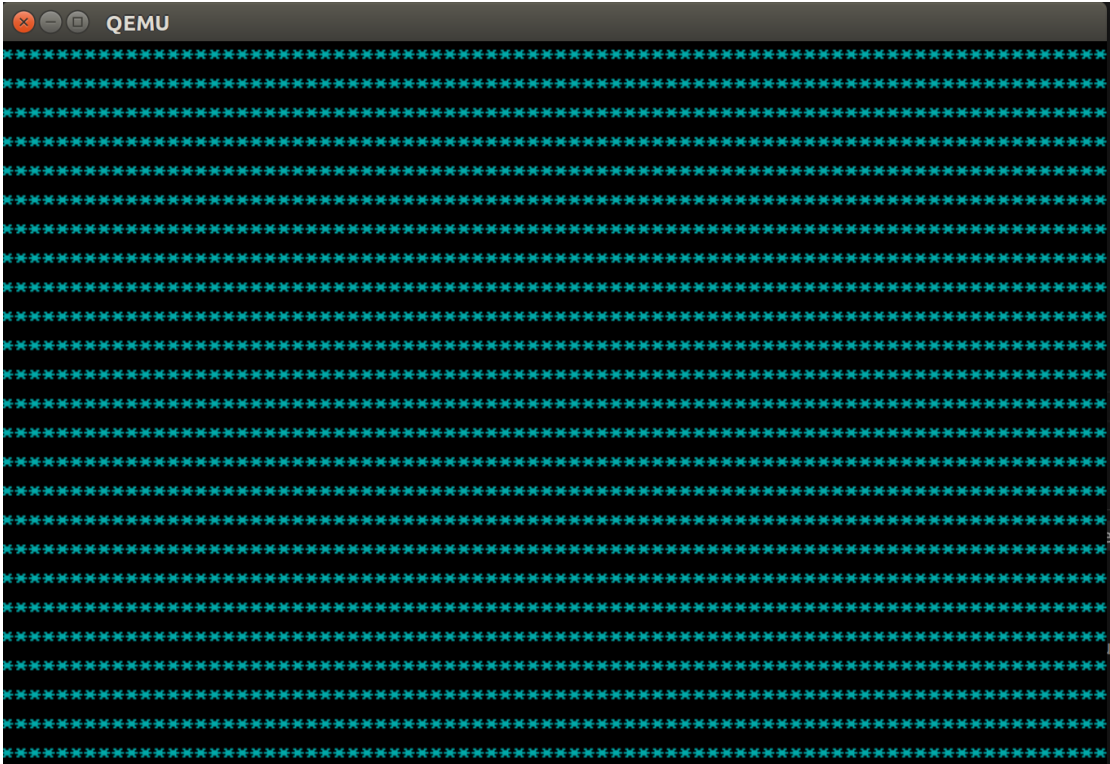
经典的启动方式是指下面的计算机启动过程是x86架构下的计算机BIOS启动过程，而UEFI启动或在arm架构下则是另外一种启动方式。

1. **加电开机。**按下电源的开关，电源马上开始向主板和它的设别开始供电。此时的电压还不是很稳定，主板上的控制芯片组会向CPU发出并保持一个reset（重置）信号，让CPU内部自动恢复到初始状态下。当芯片组检测到电源已经开始稳定的供电了，芯片组则开始撤去reset信号。此时，CPU马上开始从0xFFFF0处执行指令。这个地址位于系统的BIOS的地址范围内，其实放在这里的只是一条跳转指令，指向BIOS中真正的启动代码地方。BIOS，基本输入输出系统（Basic Input Output System），是一组固化到计算机内主板上一个ROM（Read-Only Memory）只读存储器。BIOS保存着计算机最重要的基本输入输出的程序、系统设置信息、开机上电自检程序和系统启动自检程序。
2. BIOS启动。BIOS启动后，第一件事情就是执行POST(Power-On-self-test)自检阶段，主要针对系统的一些关键设备是否存在或者是功能是否正常，如：内存、显卡等。如果在POST过程中系统设备存在致命的问题，BIOS将会发出声音来报告检测过程中出现的错误，声音的长短及次数对应着系统的错误类型。POST过程会非常快，对用户几乎感觉不出来。
3. 加载MBR。BIOS按照设定好的启动顺序，将控制权交给排在第一位的存储设备，即设备的首扇区512字节，称为MBR(Master Boot Record, 主引导扇区)，并且将这512字节复制到放在0x7c00的内存地址中运行。存储设备一般分为若干个固定大小的块来访问，这个固定大小的块被称为扇区，而第1个扇区被称为首扇区。但在复制之前，计算机会根据MBR判断设备是不是可启动的，即有无操作系统。判断依据是检查MBR最后两个字节是否为0x55,0xAA。
4. 硬盘启动。MBR只有512字节大小，程序可处理的逻辑有限。因此MBR会从存储设备中加载bootloader(启动管理器)，bootloader并无大小限制。bootloader的作用是初始化环境，然后从存储设备加载kernel(操作系统内核)到内存中。
5. **内核启动。**kernel加载入内存后，bootloader跳转到kernel处执行。至此，计算机启动完毕。

我们需要编写的内容是MBR，bootloader和kernel，而BIOS启动，POST，MBR被加载到0x7c00的过程由计算机自动完成。

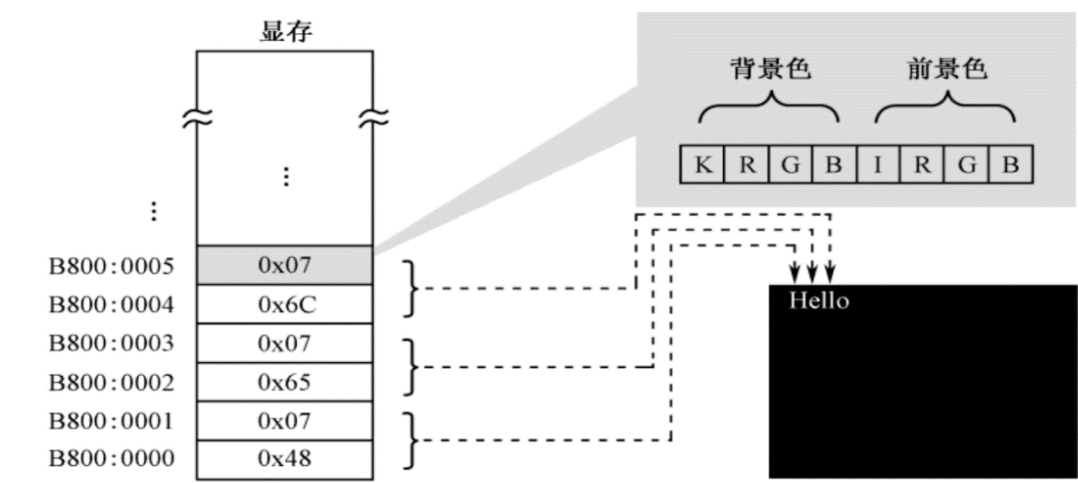
(四) 字符显示原理

1. qemu 显示屏实际上是按 25x80 个字符来排列的矩阵，如下所示：



2. 如何将字符放入显示矩阵

为了便于控制显示，IA-32 处理器将显示矩阵映射到内存地址 0xB8000~0xBFFFF 处，这段地址称为显存地址。在文本模式下，控制器的最小可控制单位为字符。每一个显示字符自上而下，从左到右依次使用 0xB8000~0xBFFFF 中的两个字节表示。其中，低字节表示显示的字符，高字节表示字符的颜色属性，如下所示



字符的颜色属性的字节高 4 位表示背景色，低 4 位表示前景色，如下所示：

R	G	B	背景色	前景色	
			K=0 时不闪烁, K=1 时闪烁	I=0	I=1
0	0	0	黑	黑	灰
0	0	1	蓝	蓝	浅蓝
0	1	0	绿	绿	浅绿
0	1	1	青	青	浅青
1	0	0	红	红	浅红
1	0	1	品(洋)红	品(洋)红	浅品(洋)红
1	1	0	棕	棕	黄
1	1	1	白	白	亮白

在上面的对显示矩阵的点的描述中，我们使用的是二维的点，但对应到显存是一维的，因此我们需要进行维度的转换，即显示矩阵的点(x,y)(x,y)对应到显存的起始位置如下所示：

$$\text{显存起始位置} = 0xB8000 + 2 \cdot (80 \cdot x + y)$$

其中，(x,y)表示第 x 行第 y 列，公式中乘 2 的原因是每个显示字符使用两个字节表示。

(五) 实模式中断

实模式下的中断非：中断向量表存放在 0x0~0x3ff 之间，每 4 字节为一个表项，每个表项的值指向一段程序的入口，这段程序就是处理中断的程序，下面为实验要用到的功能号。

功能	功能号	参数	返回值
设置光标位置	AH=02H	BH=页码, DH=行, DL=列	无
获取光标位置和形状	AH=03H	BX=页码	AX=0, CH=行扫描开始, CL=行扫描结束, DH=行, DL=列
在当前光标位置写字符和属性	AH=09H	AL=字符, BH=页码, BL=颜色, CX=输出字符的个数	无

一般地, 中断的调用方式如下。

```
将参数和功能号写入寄存器
int 中断号
从寄存器中取出返回值
```

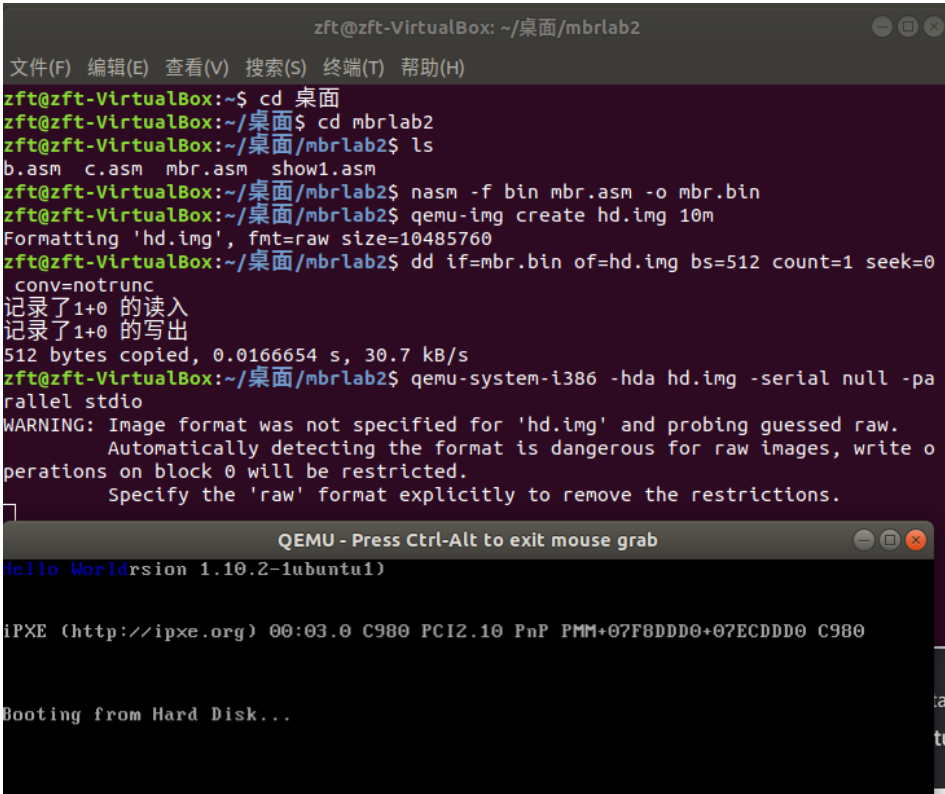
四、实验过程

(一) Assignment 1 MBR

(注意: 寄存器请使用 16 位的寄存器)

1. 复现 example 1

编写 MBR 的代码, 在 MBR 被加载到内存地址 0x7c00 后, 向屏幕输出蓝色的 Hello World。



```
zft@zft-VirtualBox: ~/桌面/mbrlab2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
zft@zft-VirtualBox:~$ cd 桌面
zft@zft-VirtualBox:~/桌面$ cd mbrlab2
zft@zft-VirtualBox:~/桌面/mbrlab2$ ls
b.asm c.asm mbr.asm show1.asm
zft@zft-VirtualBox:~/桌面/mbrlab2$ nasm -f bin mbr.asm -o mbr.bin
zft@zft-VirtualBox:~/桌面/mbrlab2$ qemu-img create hd.img 10m
Formatting 'hd.img', fmt=raw size=10485760
zft@zft-VirtualBox:~/桌面/mbrlab2$ dd if=mbr.bin of=hd.img bs=512 count=1 seek=0
conv=notrunc
记录了1+0 的读入
记录了1+0 的写出
512 bytes copied, 0.0166654 s, 30.7 kB/s
zft@zft-VirtualBox:~/桌面/mbrlab2$ qemu-system-i386 -hda hd.img -serial null -parallel stdio
WARNING: Image format was not specified for 'hd.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
QEMU - Press Ctrl-Alt to exit mouse grab
Hello Worldrsion 1.10.2-1ubuntu1)
iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F8DDDD+07ECDDDD C980
Booting from Hard Disk...
```

如图所示, 结果符合预期。

2. 修改 example 1 的代码

要求: MBR 被加载到 0x7C00 后在(12,12)处开始输出你的学号。注意, 你的学号显示的前景色和背景色必须和教程中不同。说说你是怎么做的, 并将结果截图。

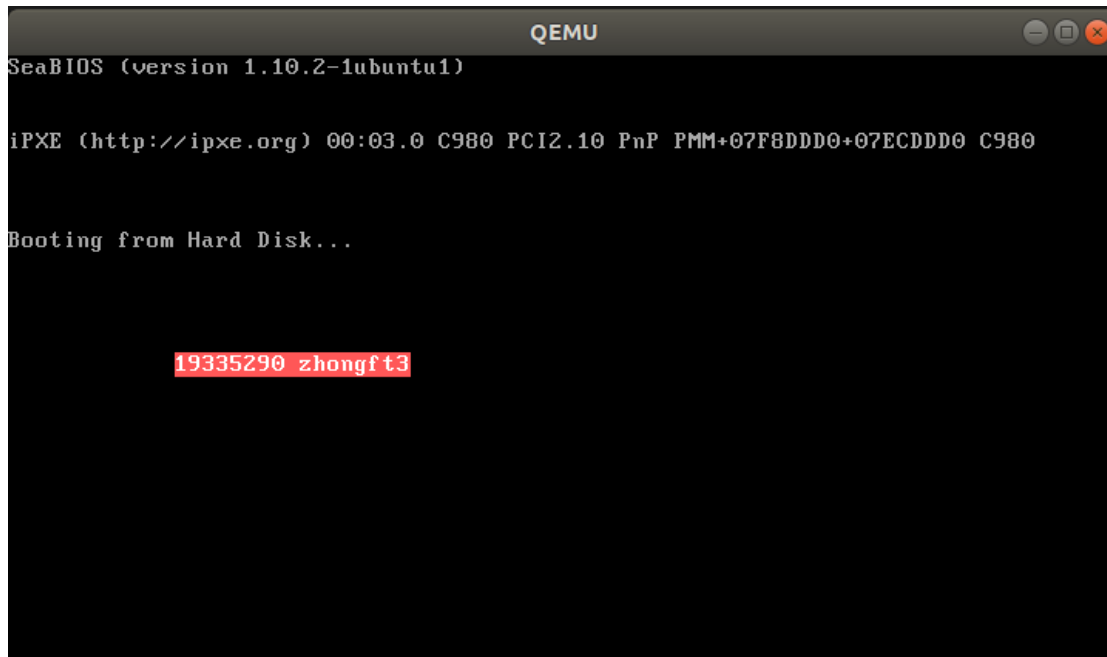
过程：由 qemu 显示屏是按 25x80 个字符来排列的矩阵可知 (12,12) 的位置怎么计算，即显存起始位置=0xB8000+2·(80·12+12)=0xB8000+2·972

```
;初始化栈指针
mov sp, 0x7c00
mov ax, 0xb800
mov gs, ax

mov ah, 0xCF; 红底白字
mov al, '1'
mov [gs:2 * 972], ax

mov al, '9'
mov [gs:2 * 973], ax

mov al, '3'
mov [gs:2 * 974], ax
```



如图所示，结果符合预期。

(二) Assignment 2 实模式中断

1. 请探索实模式下的光标中断，利用中断实现光标的位置获取和光标的移动

过程：使用了 int 10h 中断的 2 号，3 号，9 号功能。先设置光标位置，在在光标位置显示字符。下面这些图表明了过程：

```

(gdb) x/5ah $pc
=> 0x7c00: mov $0x6,%ah
0x7c02: mov $0x0,%al
0x7c04: mov $0x0,%ch
0x7c06: mov $0x0,%cl
0x7c08: mov $0x19,%dh
0x7c0a: mov $0x50,%dl
0x7c0c: int $0x10
0x7c0e: mov $0x2,%ah
0x7c10: mov $0x0,%bh
0x7c12: mov $0x1,%dh
0x7c14: mov $0x1,%dl
0x7c16: int $0x10
0x7c18: mov $0x9,%ah
0x7c1a: mov $0x61,%al
0x7c1c: mov $0xcf,%bl
0x7c1e: mov $0x0,%bh
0x7c20: mov $0x10cd0003,%ecx
0x7c22: mov $0x3,%ah
0x7c24: mov $0x0,%bh
0x7c26: int $0x10
0x7c28: mov $0x2,%ah
0x7c2a: mov $0x0,%bh
0x7c2c: add $0x1,%dh
---Type <return> to continue, or q <return> to quit---
0x7c2e: add $0x1,%dl
0x7c30: int $0x10
0x7c32: mov $0x9,%ah
0x7c34: mov $0x61,%al
0x7c36: mov $0xcf,%bl
0x7c38: mov $0x0,%bh
0x7c3a: mov $0x10cd0003,%ecx
0x7c3c: jmp 0x7c44
0x7c3e: add %al,(%eax)
0x7c40: add %al,(%eax)
0x7c42: add %al,(%eax)
0x7c44: add %al,(%eax)

```

上面这张图表示设置断点的位置

```

(gdb) b *0x7c00
Breakpoint 1 at 0x7c00
(gdb) c
Continuing.

Breakpoint 1, 0x00007c00 in ?? ()
(gdb) info registers
eax             0xaa55      43605
ecx             0x0        0
edx             0x80       128
ebx             0x0        0
esp             0x6f04     0x6f04
ebp             0x0        0x0
esi             0x0        0
edi             0x0        0
eip             0x7c00     0x7c00
eflags          0x202     [ IF ]
cs              0x0        0
ss              0x0        0
ds              0x0        0
es              0x0        0
fs              0x0        0
gs              0x0        0

```

这是一开始寄存器的值

下面几张图片表明了光标的位置,可以通过观察寄存器 edx 的值知道与程序设计相符

```

1  org 0x7c00 ; 程序装载到 7C00h, 这! Continuing.
2
3  mov ah,6 ; 清屏
4  mov al,0
5  mov ch,0
6  mov cl,0
7  mov dh,25
8  mov dl,80
9  int 10h
10
11 mov ah,2 ; 设置光标位置
12 mov bh,0 ; 第0页
13 mov dh,1 ; dh放行号, dl放列号
14 mov dl,1
15 int 10h
16
17 mov ah,9 ; 在光标位置显示字符串
18 mov al,'a'
19 mov bl,0xCF ; 红底白字
20 mov bh,0 ; 第0页
21 mov cx,3 ; 字符串个数

```

```

Breakpoint 2, 0x00007c0c in ?? ()
(gdb) info registers
Undefined info command: "registera". Try "help info".
(gdb) info registers
eax             0x0000     1536
ecx             0x0        0
edx             0x1950     6480
ebx             0x0        0
esp             0x6f04     0x6f04
ebp             0x0        0x0
esi             0x0        0
edi             0x0        0
eip             0x7c0c     0x7c0c
eflags          0x202     [ IF ]
cs              0x0        0
ss              0x0        0
ds              0x0        0
es              0x0        0
fs              0x0        0
gs              0x0        0

```



```

(gdb) c
Continuing.

Breakpoint 3, 0x00007c16 in ?? ()
(gdb) info registers
eax      0x200    512
ecx      0x0      0
edx      0x101    257
ebx      0x0      0
esp      0x6f04   0x6f04
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x7c16   0x7c16
eflags   0x202    [ IF ]
cs       0x0      0
ss       0x0      0
ds       0x0      0
es       0x0      0
fs       0x0      0
gs       0x0      0

mov ah,6 ;清屏
mov al,0
mov ch,0
mov cl,0
mov dh,25
mov dl,80
int 10h

mov ah,2 ;设置光标位置
mov bh,0;第0页
mov dh,1;dh放行号, dl放列号
mov dl,1
int 10h

mov ah,9;在光标位置显示字符串
mov al,'a'
mov bl,0xCF ;红底白字
mov bh,0;第0页
mov cx,3 ;字符串个数

```

```

16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620

```

```
(gdb) c
Continuing.

Breakpoint 7, 0x00007c44 in ?? ()
(gdb) info registers
eax            0x961      2401
ecx            0x3        3
edx            0x202      514
ebx            0xcf       207
esp            0x6f04     0x6f04
ebp            0x0        0x0
esi            0x0        0
edi            0x0        0
eip            0x7c44     0x7c44
eflags         0x202      [ IF ]
cs             0x0        0
ss             0x0        0
ds             0x0        0
es             0x0        0
fs             0x0        0
gs             0x0        0
```

aaa
aaa

实验符合预期

2. 修改 Assignment 1 的代码，使用实模式下的中断来输出你的学号

过程：使用了 int 10h 中断的 13H 功能。先设置光标位置，在在光标位置显示字符

```
mov ax,0x7c0
mov dx,ax
mov es,ax

mov ah,6      ;清屏
mov al,0
mov ch,0
mov cl,0
mov dh,25
mov dl,80
int 10h

mov ax,str
mov bp,ax
mov ah,13H ;在光标位置显示字符串
mov al,0
mov bh,0 ;第0页
mov bl,0xCF ;红底白字
mov cx,length
mov dh,8
mov dl,8
int 10h

jmp $ ;死循环

str:db "19335290 zhongfangting"
```

19335290 zhongfangting

3. 在 1 和 2 的知识的基础上，探索实模式的键盘中断，利用键盘中断实现键盘输入并回显

过程：使用了 int 16h 中断的 0 号功能。它会将键盘输入的字符的 ASCII 码放在寄存器 al 中，设置寄存器 ah 的颜色，将 ax 复制给 0xB8000，就会在 qemu 屏幕的第一个位置显示输入的字符。结果如下：输入字符 a，qemu 屏幕上显示 a.

```
mov gs, ax
; 初始化栈指针
mov sp, 0x7c00
mov ax, 0xb800
mov gs, ax

start:
mov ah, 6      ; 清屏
mov al, 0
mov ch, 0
mov cl, 0
mov dh, 25
mov dl, 80
int 10h
mov ah, 0; 调用16号中断的0号功能输入字符
int 16h
; mov ah, 01H; 调用16号中断的1号功能
; int 16h
mov ah, 0XCF ; 红底白字, 输出刚才输入的字符
mov [gs:2*0], ax

jmp $ ; 死循环
times 510 - ($ - $$) db 0
db 0x55, 0xaa
```

(三) Assignment 3 汇编

1. 分支逻辑的实现

请将下列伪代码转换成汇编代码，并放置在标号 your_if 之后

• 最后附上 make run 的截图，并

3.1 分支逻辑的实现

请将下列伪代码转换成汇编代码，并

```
if a1 < 12 then
  if_flag = a1 / 2 + 1
else if a1 < 24 then
  if_flag = (24 - a1) * a1
else
  if_flag = a1 << 4
end
```

3.2 循环逻辑的实现

请将下列伪代码转换成汇编代码，并

```
while a2 >= 12 then
  call my_random      // my_random
  while_flag[a2 - 12] = eax
  --a2
end
```

```
6 your_if:
7 mov edx,[a1]
8 cmp edx,12
9 jl smaller_than_12
10 cmp edx,24
11 jl smaller_than_24
12 jmp larger_than_24
13
14 smaller_than_12: ;a1<12
15 mov eax,[a1]
16 mov ebx,2
17 idiv ebx
18 inc eax
19 mov [if_flag],eax
20 jmp end
21
22 smaller_than_24: ;12<=a1<24
23 mov eax,24
24 sub eax,[a1]
25 imul eax,edx
26 mov [if_flag],eax
27 jmp end
28
29 larger_than_24: ;a1>=24
30 mov eax,[a1]
31 shl eax,4
32 mov [if_flag],eax
33 jmp end
34
35 end:
```

2. 循环逻辑的实现

请将下列伪代码转换成汇编代码，并放置在标号 your_while 之后

```
if_flag = (24 - a1) * a1
else
    if_flag = a1 << 4
end

3.2 循环逻辑的实现
请将下列伪代码转换成汇编代码，并放置在标号 your_while 之后。

while a2 >= 12 then
    call my_random // my_random将产生一个随机数放到eax中返回
    while_flag[a2 - 12] = eax
    --a2
end

; put your implementation here
your_while:
loop1:
    mov edx,[a2]
    cmp edx,12
    jl end_loop
    call my_random
    mov ebx,[while_flag]
    ;!!!! because the value of edx had been changed while exe cm
    mov edx,[a2]
    mov byte[ebx+(edx-12)],al
    dec edx
    mov [a2],edx
    jmp loop1
end_loop:
```

3. 函数的实现

请编写函数 your_function 并调用之，函数的内容是遍历字符串 string

```
call my_random // my_random将产生一个随机数
while_flag[a2 - 12] = eax
--a2
end

3.3 函数的实现
请编写函数 your_function 并调用之，函数的内容是遍历字符串 string

your_function:
    for i = 0; string[i] != '\0'; ++i then
        pushad
        push string[i] to stack
        call print_a_char
        popad
    end
    return
end

#include "end.include"
your_function:
; put your implementation here
xor eax,eax
xor ecx,ecx
mov ebx,[your_string]
loop2:
    mov cl,byte[ebx+ecx]; 字节对应寄存器cl, 而不是ecx
    inc ecx
    cmp ecx,0
    je end_for
    pushad
    push ecx
    call print_a_char
    pop ecx
    popad
    jmp loop2
end_for:
ret
```

运行结果如下：

```
zft@zft-Virtual Box: ~/桌面/assignment$ make run
>>> begin test
>>> if test pass!
>>> while test pass!
Mr. Chen, students and TAs are the best!
zft@zft-Virtual Box: ~/桌面/assignment$
```

(四) Assignment 4 汇编小程序

字符弹射程序。请编写一个字符弹射程序，其从点(2,0)处开始向右下角 45 度开始射出，遇到边界反弹，反弹后按 45 度角射出，方向视反弹位置而定。同时，你可以加入一些其他效果，如变色，双向射出等。注意，你的程序应该不超过 510 字节，否则无法放入 MBR 中被加载执行。

过程：首先要写一个函数来延时，控制画框显示速度和抖动速度；接下来写函数判断字符该往哪个方向走并打印字符。

延时函数如下：

```
Dn_Rt equ 1 ; D-Down,U-Up,R-right,L-Left
Up_Rt equ 2
Up_Lt equ 3
Dn_Lt equ 4
delay equ 1000 ; 计时器延迟计数,用于控制画框的速度
ddelay equ 100 ; 计时器延迟计数,用于控制画框的速度

org 7C00h ; 程序装载到7C00h, 这里存放的是主引导记录
```

```

dec word[count]    ;递减计数变量
jnz loop1          ;>0: 跳转;
mov word[count],delay
dec word[dcount]    ;递减计数变量
jnz loop1
mov word[count],delay ;延时
mov word[dcount],ddelay

```

判断字符往哪个方向走:
(只贴出了一部分)

```

mov al,1
cmp al,byte[rdul]
jz DnRt
mov al,2
cmp al,byte[rdul]
jz UpRt
mov al,3
cmp al,byte[rdul]
jz UpLt
mov al,4
cmp al,byte[rdul]
jz DnLt

```

```

dr2ur:
mov word[x],paddingh
mov byte[rdul],Up_Rt
jmp show

dr2dl:
mov word[y],paddingw
mov byte[rdul],Dn_Lt
jmp show

```

打印字符:

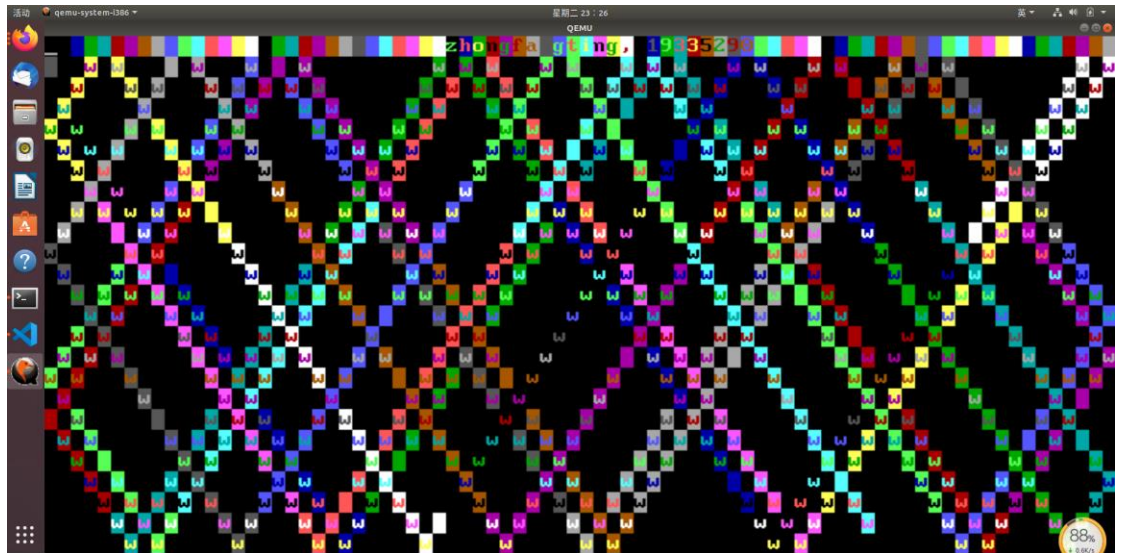
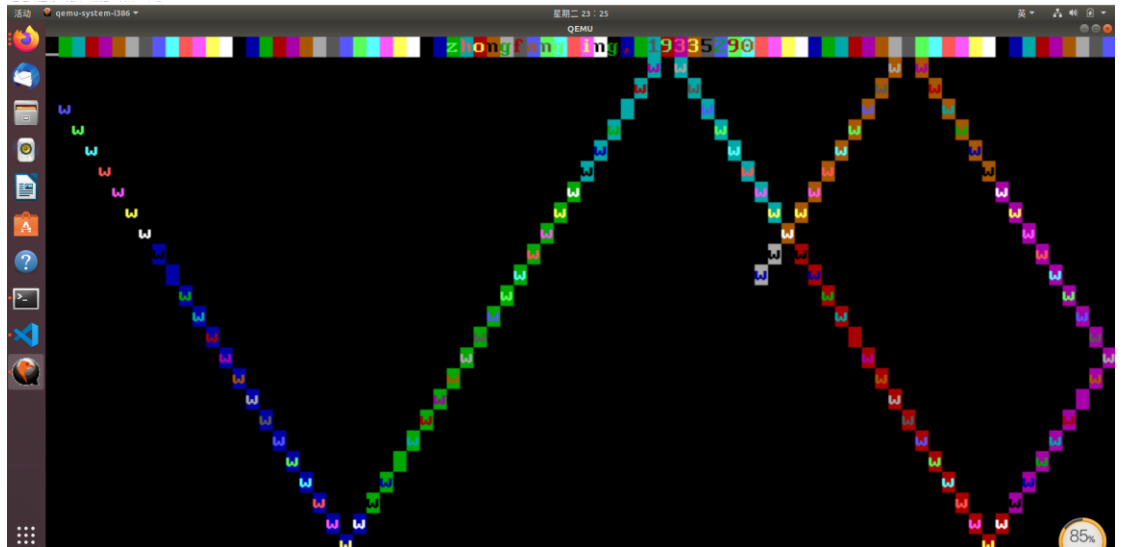
```

show:
xor ax,ax          ;计算显存地址
mov ax,word[x]
mov bx,80
mul bx
add ax,word[y]
mov bx,2
mul bx
mov bp,ax
mov ah,[curcolor2] ;弹字符的背景色和前景色 (默认值为07h, 详见文档)
inc byte[curcolor2]
cmp byte[curcolor2], 0fh
jnz skip
;mov byte[curcolor2], 1 ;为了不改变背景色
skip:
mov al,byte[char]   ;AL = 显示字符值 (默认值为20h=空格符)
mov word[gs:bp],ax  ;显示字符的ASCII码值

mov si, myinfo      ;显示姓名和学号
mov di, 2
mov cx, word[infoLen]
loop2:              ;显示myinfo中的每个字符
mov al, byte[ds:si]
inc si
mov ah, [curcolor]  ;背景色和前景色
add byte[curcolor], 12h ;变色, 该数字可随意调节以达到不错的效果
mov word[es:di],ax

```

结果如下:



五、 实验心得

这次实验做得比较艰难，因为对 8086 这门语言不熟悉，在写汇编程序上耗费了许多时间。

1. 我一开始不知道怎么使用 gdb 调试程序，尤其是在连上 qemu 后，输入指令后显示“没有加载符号表”错误，这样我怀疑自己是不是没连上 qemu，在网上也查不到解决方案。后来问了助教，说这个不影响，又问了同学才知道怎么调试。
2. 在 assignment 2 中，我对中断查了很多资料，知道了它们的功能，但对怎么用它们还是云里雾里的。尤其是键盘中断，通过使用 int 16h 的 0 号功能输入字符，但怎么将这个字符在 qemu 上显示这里出了问题，无法通过 int 10h 中断来输出字符，按理说寄存器 al 已经存入字符的 ASCII 码，可以调用 int 10h 来进行显示字符。但我不行，最后只好用了 assignment 1 的方法来进行字符的显示。

3. 在 assignment 3 中，我也花了不少时间，首先 a1,a2 我没有赋给寄存器直接使用犯了错。其次在第 3 个函数遍历字符串数组出了很多错，最后一点点排查，才将其改完。
4. 在 assignment 4 中，我觉得难点在判断字符碰壁后朝哪个方向运动。这个也花了我很长时间。

总之，我觉得这次实验收获非常大。