



本科生实验报告

实验课程：_____操作系统_____

实验名称：_____编译内核_____

专业名称：_____计算机科学与技术_____

学生姓名：_____凌国明_____

学生学号：_____21307077_____

实验地点：_____教室_____

实验成绩：_____

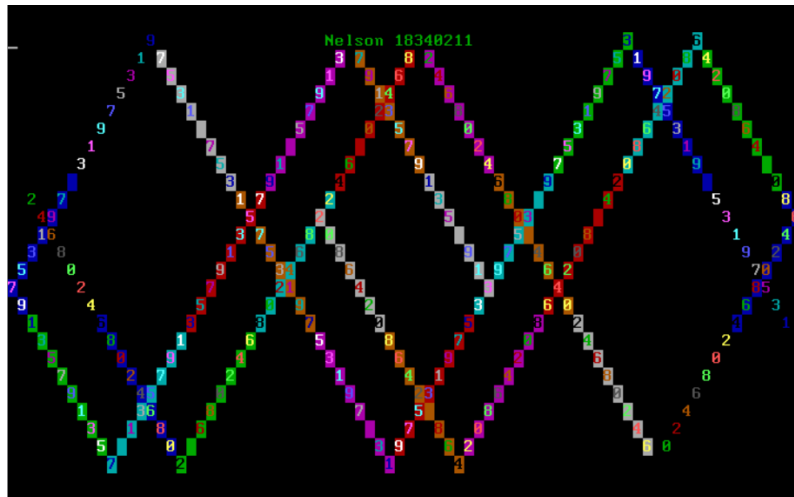
报告时间：_____2023. 03. 22_____

1. 实验要求

- (1) 学习 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。
- (2) 根据所学的知识，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。
- (3) 将学习如何使用 gdb 来调试程序的基本方法。

实验任务

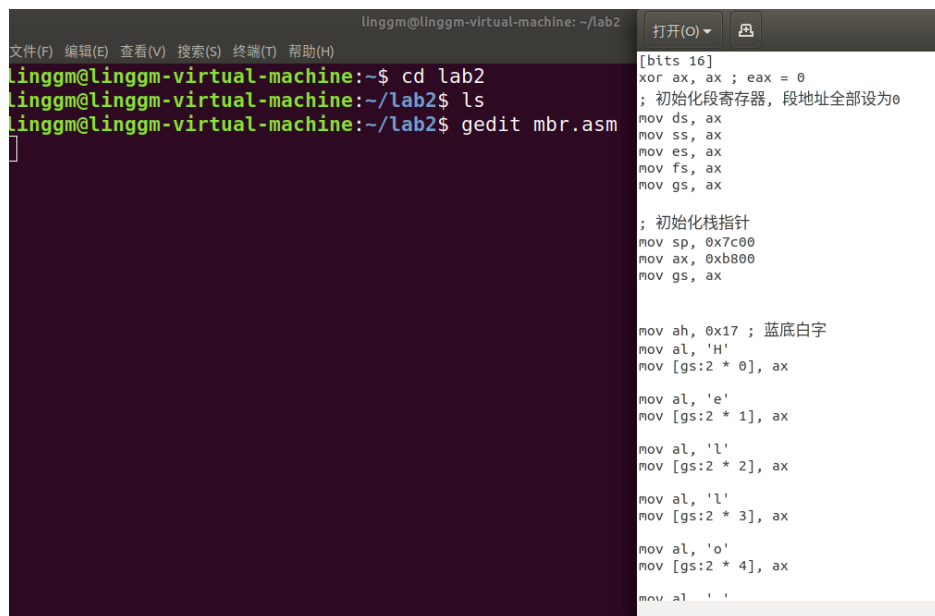
- (1) (i) **复现 example 1**。说说你是怎么做的，并将结果截图。
(ii) 请修改 example 1 的代码，使得 MBR 被加载到 0x7C00 后在(12,12)(12,12)处开始**输出你的学号**。注意，你的学号显示的前景色和背景色必须和教程中不同。说说你是怎么做的，并将结果截图。
- (2) (i) 请探索实模式下的光标中断，**利用中断实现光标的位置获取和光标的移动**。说说你是怎么做的，并将结果截图。
(ii) 请修改 1.2 的代码，**使用实模式下的中断来输出你的学号**。说说你是怎么做的，并将结果截图
(iii) 在 2.1 和 2.2 的知识的基础上，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**
- (3) 用 x86 汇编语言实现**分支逻辑，循环逻辑和函数调用**，并通过测试
- (4) 用 x86 汇编语言编写**字符弹射小程序**。其从点(2,0)(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。



2. 实验过程

2.1.1 复现 example1:

- 第一步：编写 mbr



```
linggm@linggm-virtual-machine: ~/lab2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
linggm@linggm-virtual-machine:~$ cd lab2
linggm@linggm-virtual-machine:~/lab2$ ls
linggm@linggm-virtual-machine:~/lab2$ gedit mbr.asm

[bits 16]
xor ax, ax ; eax = 0
; 初始化段寄存器, 段地址全部设为0
mov ds, ax
mov ss, ax
mov es, ax
mov fs, ax
mov gs, ax

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

mov ah, 0x17 ; 蓝底白字
mov al, 'H'
mov [gs:2 * 0], ax

mov al, 'e'
mov [gs:2 * 1], ax

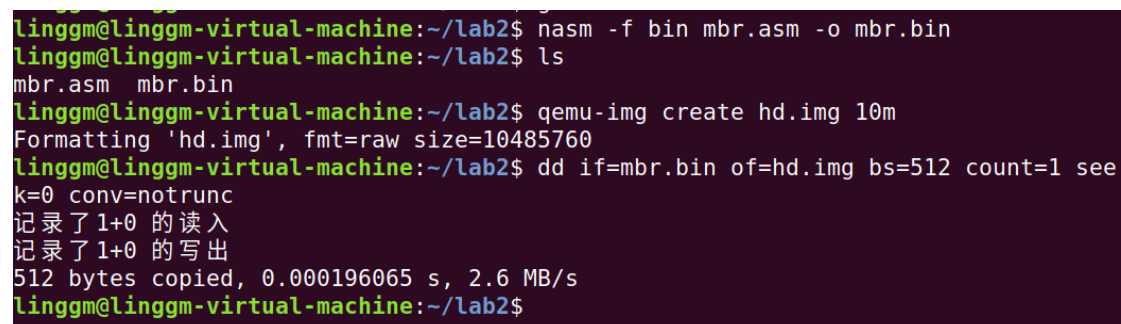
mov al, 'l'
mov [gs:2 * 2], ax

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

mov al, 'o'
mov [gs:2 * 4], ax

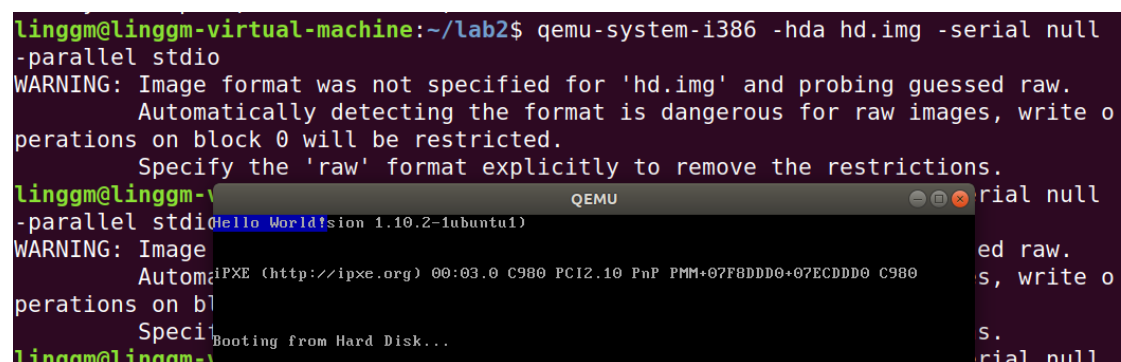
mov al, ' '
```

- 第二步：将 mbr.asm 汇编成 bin 格式；
创建磁盘；将 mbr.bin 写入磁盘首扇区



```
linggm@linggm-virtual-machine:~/lab2$ nasm -f bin mbr.asm -o mbr.bin
linggm@linggm-virtual-machine:~/lab2$ ls
mbr.asm mbr.bin
linggm@linggm-virtual-machine:~/lab2$ qemu-img create hd.img 10m
Formatting 'hd.img', fmt=raw size=10485760
linggm@linggm-virtual-machine:~/lab2$ dd if=mbr.bin of=hd.img bs=512 count=1 see
k=0 conv=notrunc
记录了1+0 的读入
记录了1+0 的写出
512 bytes copied, 0.000196065 s, 2.6 MB/s
linggm@linggm-virtual-machine:~/lab2$
```

- 第三步：启动 qemu 进行测试



```
linggm@linggm-virtual-machine:~/lab2$ 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 o
perations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
linggm@linggm-virtual-machine:~/lab2$
QEMU
Hello World!
WARNING: Image
Automat
iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F8DD0+07ECDD0 C980
perations on bl
Specifi
Booting from Hard Disk...
linggm@linggm-virtual-machine:~/lab2$
```

至此，example1 复现完毕，启动 gdb 调试可参考任务 1.2。

2.1.2 修改 example 1 的代码，使得 MBR 被加载到后输出我的学号

- 第一步：编写 mbr

```
linggm@linggm-virtual-machine: ~/lab2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
linggm@linggm-virtual-machine:~$ cd lab2
linggm@linggm-virtual-machine:~/lab2$ ls
hd.img  mbr.asm
linggm@linggm-virtual-machine:~/lab2$ gedit mbr.asm
mov gs, ax
mov ah, 0x17 ; 蓝底白字
mov al, '2'
mov [gs:2 * (12 * 80 + 12)], ax
mov al, '1'
mov [gs:2 * (12 * 80 + 13)], ax
mov al, '3'
mov [gs:2 * (12 * 80 + 14)], ax
mov al, '0'
mov [gs:2 * (12 * 80 + 15)], ax
mov al, '7'
mov [gs:2 * (12 * 80 + 16)], ax
mov al, '0'
mov [gs:2 * (12 * 80 + 17)], ax
mov al, '7'
mov [gs:2 * (12 * 80 + 18)], ax
mov al, '7'
mov [gs:2 * (12 * 80 + 19)], ax
jmp $ ; 死循环
times 510 - ($ - $$) db 0
db 0x55, 0xaa
```

- 第二步：汇编为 elf32，链接，生成符号表

```
linggm@linggm-virtual-machine:~/lab2$ gedit mbr.asm
linggm@linggm-virtual-machine:~/lab2$ nasm -f elf32 -o mbr.o mbr.asm
linggm@linggm-virtual-machine:~/lab2$ ld -m elf_i386 -Ttext 0x7c00 -o mbr.symbol
-N mbr.o
ld: 警告：无法找到项目符号 _start; 缺省为 00000000000007c00
linggm@linggm-virtual-machine:~/lab2$ ls
hd.img  mbr.asm  mbr.o  mbr.symbol
```

- 第三步：汇编为 bin，写入磁盘首扇区

```
linggm@linggm-virtual-machine:~/lab2$ nasm -f bin -o mbr.bin mbr.asm
linggm@linggm-virtual-machine:~/lab2$ ls
hd.img  mbr.asm  mbr.bin  mbr.o  mbr.symbol
linggm@linggm-virtual-machine:~/lab2$ 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, 0.000165309 s, 3.1 MB/s
```

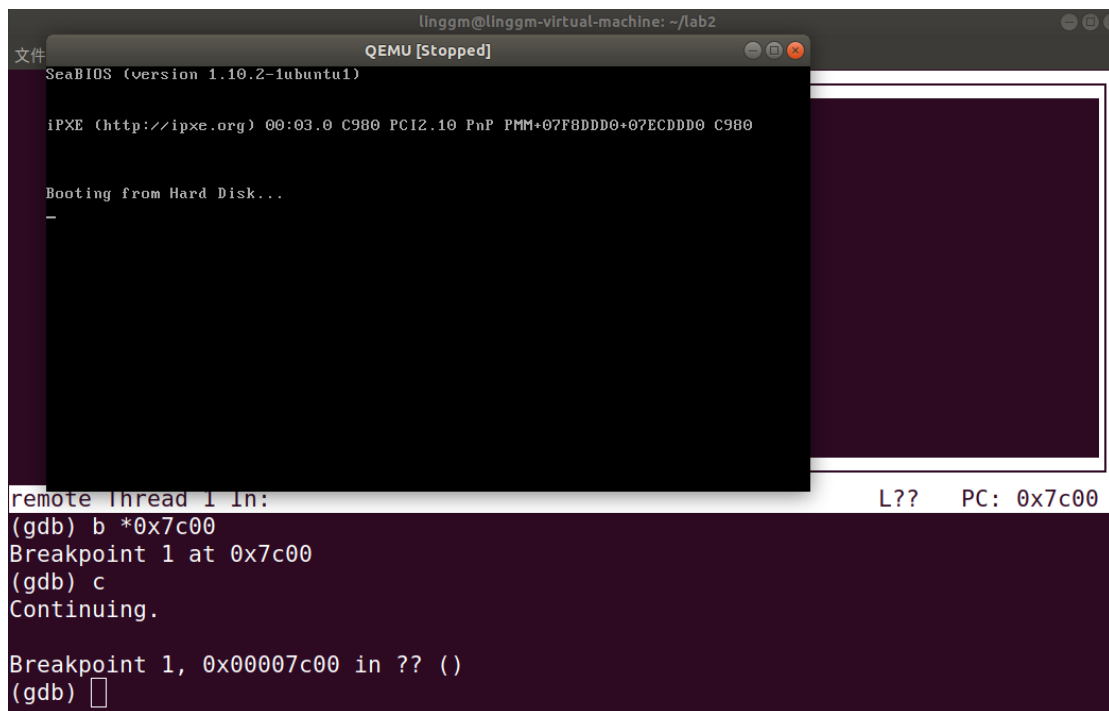
- 第四步：启动 qemu 进行测试

```
linggm@linggm-virtual-machine:~/lab2$ qemu-system-i386 -hda hd.img -s -S -serial
null -parallel stdio
```

- 第五步：启动 gdb，连接 qemu，加载符号表

```
(gdb) target remote :1234
Remote debugging using :1234
warning: No executable has been specified and target does not support
determining executable automatically. Try using the "file" command.
0x0000ffff in ?? ()
(gdb) add-symbol-file mbr.symbol 0x7c00
add symbol table from file "mbr.symbol" at
.text_addr = 0x7c00
(y or n) y
Reading symbols from mbr.symbol...(no debugging symbols found)...done.
(gdb)
```

- 第六步，设置断点，c，调试



```
linggm@linggm-virtual-machine: ~/lab2
QEMU [Stopped]
SeaBIOS (version 1.10.2-1ubuntu1)

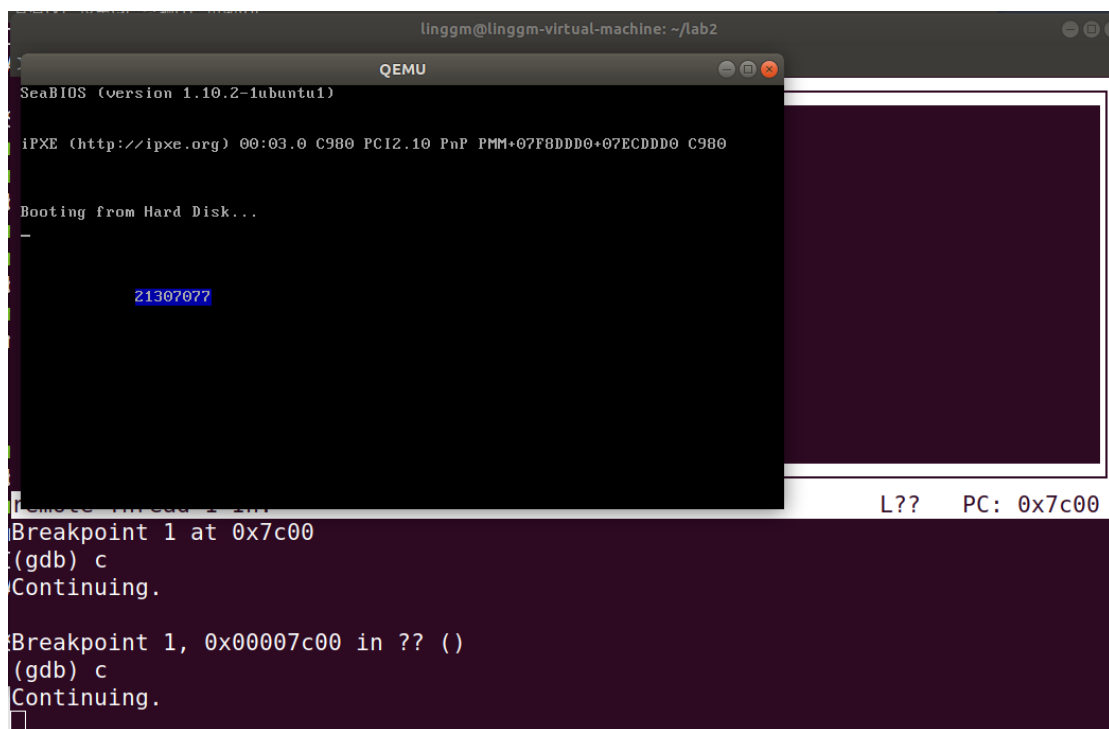
iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F8DDDD+07ECDDDD C980

Booting from Hard Disk...

remote thread 1 in: L?? PC: 0x7c00
(gdb) b *0x7c00
Breakpoint 1 at 0x7c00
(gdb) c
Continuing.

Breakpoint 1, 0x00007c00 in ?? ()
(gdb) 
```

- 到达断点后按 c 继续运行，成功显示学号，进入死循环



```
linggm@linggm-virtual-machine: ~/lab2
QEMU
SeaBIOS (version 1.10.2-1ubuntu1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F8DDDD+07ECDDDD C980

Booting from Hard Disk...

21307077

remote thread 1 in: L?? PC: 0x7c00
Breakpoint 1 at 0x7c00
(gdb) c
Continuing.

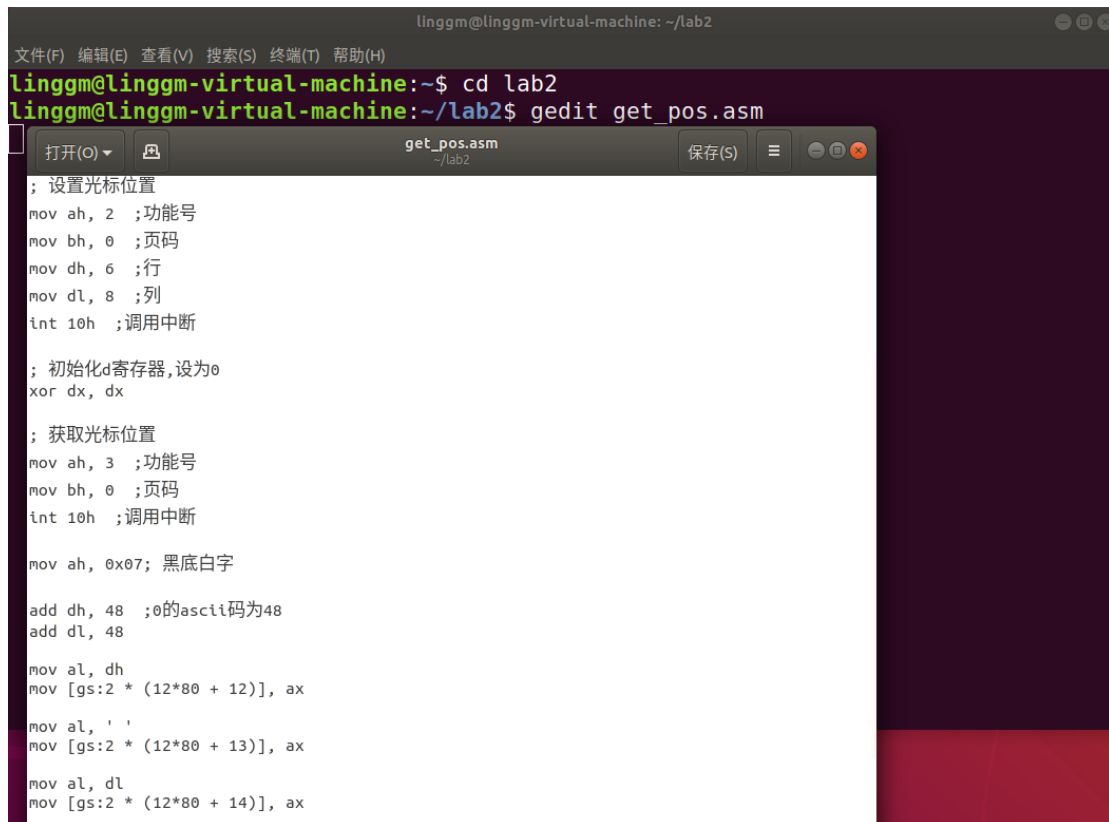
Breakpoint 1, 0x00007c00 in ?? ()
(gdb) c
Continuing.


```

至此，1.2 显示学号的任务完成

2.2.1 利用 int10h 中断，设置光标位置，然后获取光标位置

- 第一步：编写汇编程序



```
linggm@linggm-virtual-machine: ~/lab2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
linggm@linggm-virtual-machine:~$ cd lab2
linggm@linggm-virtual-machine:~/lab2$ gedit get_pos.asm

get_pos.asm
~/lab2

; 设置光标位置
mov ah, 2 ;功能号
mov bh, 0 ;页码
mov dh, 6 ;行
mov dl, 8 ;列
int 10h ;调用中断

; 初始化d寄存器,设为0
xor dx, dx

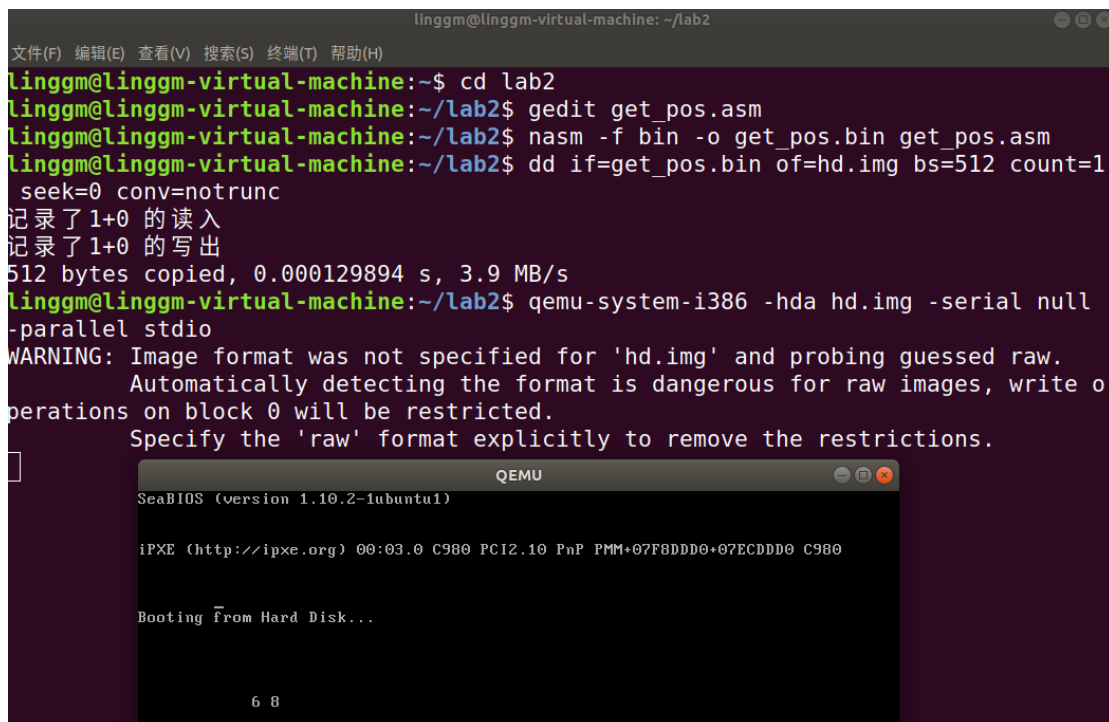
; 获取光标位置
mov ah, 3 ;功能号
mov bh, 0 ;页码
int 10h ;调用中断

mov ah, 0x07; 黑底白字

add dh, 48 ;0的ascii码为48
add dl, 48

mov al, dh
mov [gs:2 * (12*80 + 12)], ax
mov al, ' '
mov [gs:2 * (12*80 + 13)], ax
mov al, dl
mov [gs:2 * (12*80 + 14)], ax
```

- 第二步，按照 example1 的过程，将程序载入磁盘，启动 qemu 测试



```
linggm@linggm-virtual-machine: ~/lab2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
linggm@linggm-virtual-machine:~$ cd lab2
linggm@linggm-virtual-machine:~/lab2$ gedit get_pos.asm
linggm@linggm-virtual-machine:~/lab2$ nasm -f bin -o get_pos.bin get_pos.asm
linggm@linggm-virtual-machine:~/lab2$ dd if=get_pos.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, 0.000129894 s, 3.9 MB/s
linggm@linggm-virtual-machine:~/lab2$ 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 o
perations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.

QEMU
SeaBIOS (version 1.10.2-1ubuntu1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP FMM+07F8DDDD+07ECDDDD C980

Booting from Hard Disk...

6 8
```

成功设置光标行号为 6，列号为 8。成功获取光标位置并输出成功。

2.2.2 利用 int10h 中断，打印我的学号

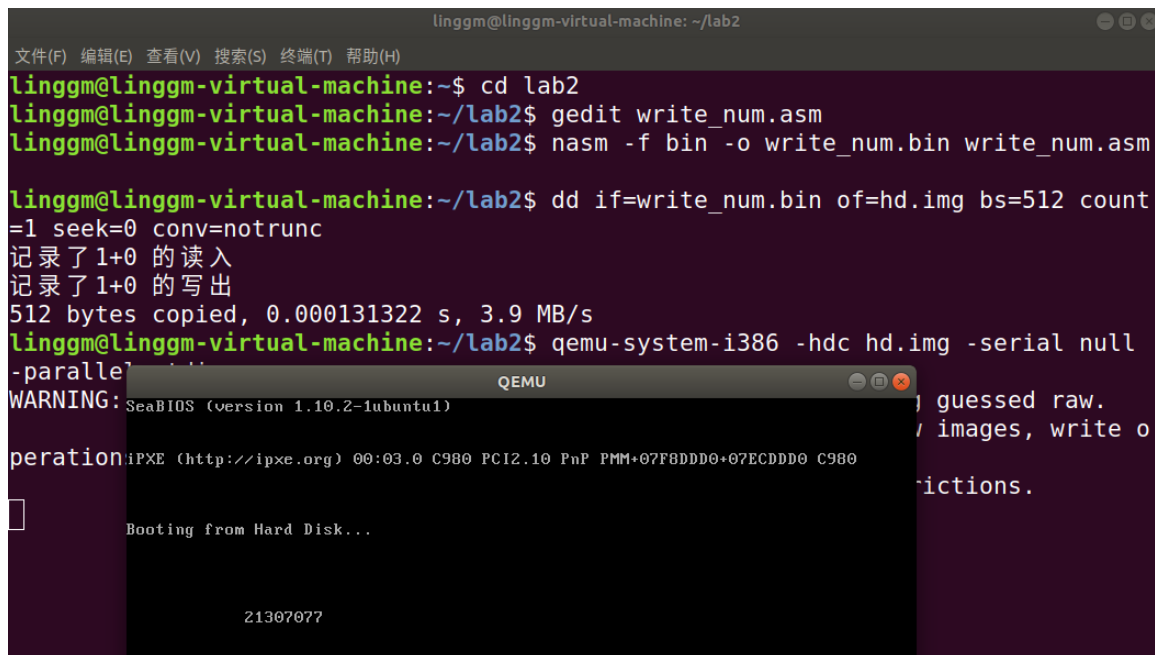
- 第一步：编写汇编程序（完整代码见 3.2 代码部分）

```
; 函数功能：将al处的字符打印在当前光标处
print_al:
    ;要求打印的字符在al处
    mov bl, 0x07 ;黑底白字
    mov bh, 0x00 ;页码为0
    mov cx, 1 ;输出一个字符
    mov ah, 0x09 ;写字符的功能号
    int 10h ;中断
    call cursor_inc
    ret

; 函数功能：当前光标位置+=1
cursor_inc:
    ; 设置光标位置
    mov ah, 2 ;功能号
    mov bh, 0 ;页码
    cmp dl, 79
    jne label ;如果dl的值不等于79，则跳转到label
    add dh, 1 ;如果dl的值等于79，则行号+1
    mov dl, -1 ;如果dl的值等于79，则列号变为-1，经下条指令变为0
label: add dl, 1 ;列坐标+1
    int 10h ;调用中断
    ret

times 510 - ($ - $$) db 0
db 0x55, 0xaa
```

- 第二步，按照 example1 的过程，将程序载入磁盘，启动 qemu 测试



```
linggm@linggm-virtual-machine: ~/lab2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
linggm@linggm-virtual-machine:~$ cd lab2
linggm@linggm-virtual-machine:~/lab2$ gedit write_num.asm
linggm@linggm-virtual-machine:~/lab2$ nasm -f bin -o write_num.bin write_num.asm
linggm@linggm-virtual-machine:~/lab2$ dd if=write_num.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, 0.000131322 s, 3.9 MB/s
linggm@linggm-virtual-machine:~/lab2$ qemu-system-i386 -hdc hd.img -serial null
-parallel
WARNING: SeaBIOS (version 1.10.2-1ubuntu1)
operation IPXE (http://ipxe.org) 00:03.0 C980 PC12.10 PnP PMM+07F8DDDD+07ECDDDD C980
guessed raw.
/ images, write o
rictions.

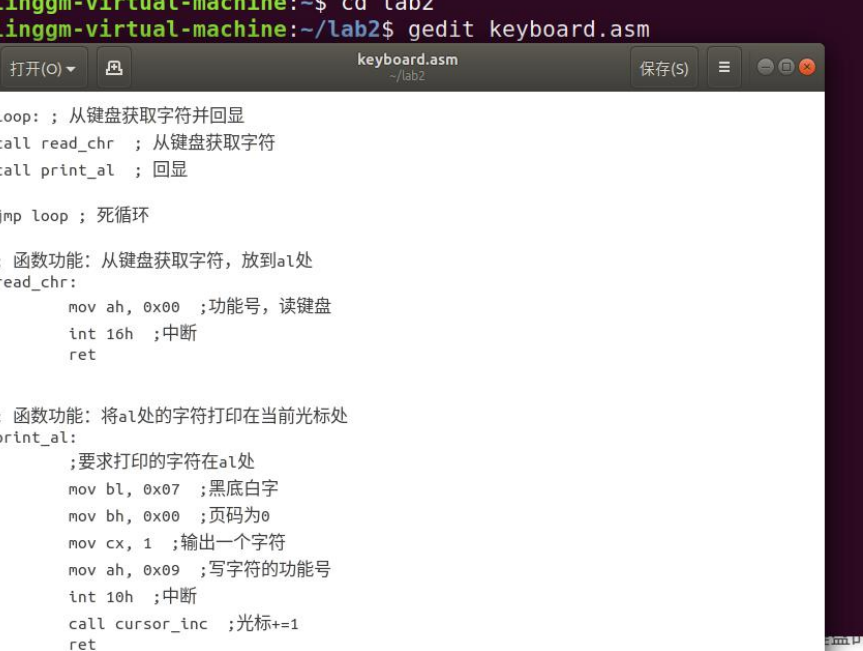
Booting from Hard Disk...

21307077
```

成功利用 int10h 中断，在 (12, 12) (12, 12) 处输出我的学号 21307077

2.2.3 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

- 第一步：编写汇编程序，完整程序及解释见 3.2 代码部分



The screenshot shows a Linux environment with a terminal window and a text editor. The terminal window at the top shows the user 'linggm' at the 'linggm-virtual-machine' prompt, navigating to the 'lab2' directory and opening 'keyboard.asm' with 'gedit'. The text editor window, titled 'keyboard.asm', displays the following assembly code:

```
loop: ; 从键盘获取字符并回显
call read_chr ; 从键盘获取字符
call print_al ; 回显

jmp loop ; 死循环

; 函数功能: 从键盘获取字符, 放到al处
read_chr:
    mov ah, 0x00 ; 功能号, 读键盘
    int 16h ; 中断
    ret

; 函数功能: 将al处的字符打印在当前光标处
print_al:
    ; 要求打印的字符在al处
    mov bl, 0x07 ; 黑底白字
    mov bh, 0x00 ; 页码为0
    mov cx, 1 ; 输出一个字符
    mov ah, 0x09 ; 写字符的功能号
    int 10h ; 中断
    call cursor_inc ; 光标+=1
    ret

; 函数功能: 当前光标位置+=1
cursor_inc:
    ; 设置光标位置
```

- 第二步，按照 example1 的过程，将程序载入磁盘，启动 qemu 测试

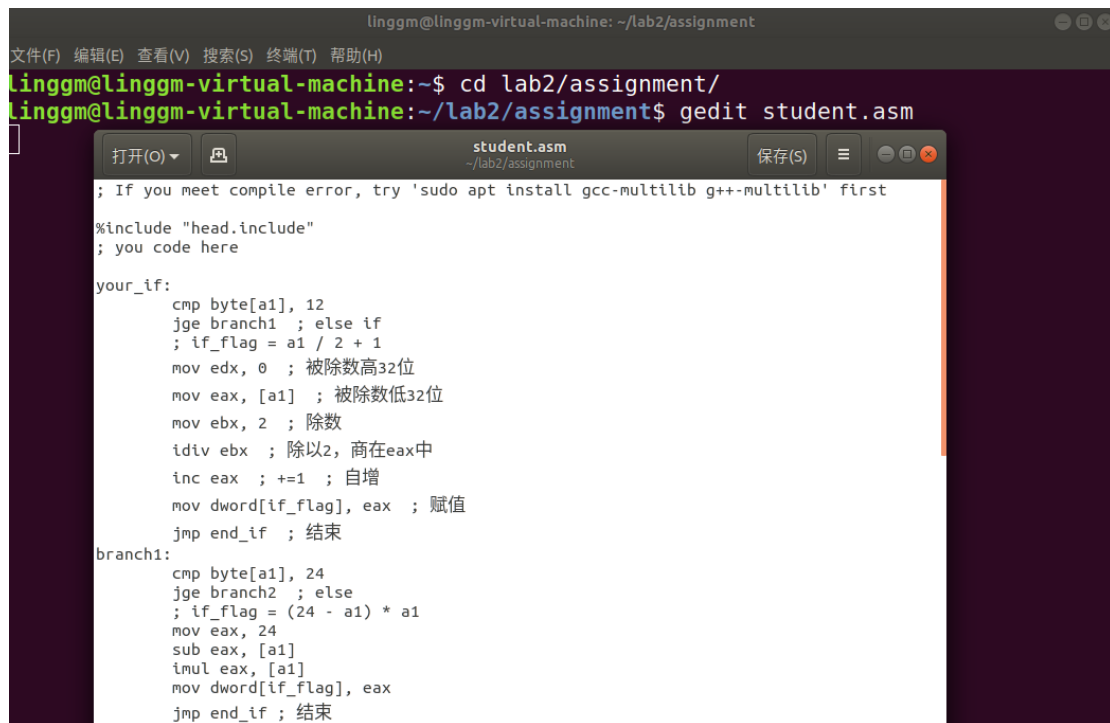
```
hggm-virtual-machine:~/lab2$ dd if=keyboard.bin of=hd.img
conv=notrunc
的读入
的写出
copied, 0.000141349 s, 3.6 MB/s
hggm-virtual-machine:~/lab2$ qemu-system-i386 -hdc hd.img
S
QEMU
SeaBIOS (version 1.10.2-1ubuntu1)
AU
C
iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F8DDDD+07ECDDDD C980
Sp
t
Booting from Hard Disk...

abcdshhfwihuasndahfuiewfuabsfuawbfa iudbwuefbiaubcasubdfuiasdifrenoigafhioHF IEWOH
FIAHIAHFUFOAFBuhuhgu
```

实现了从一直从键盘读字符，并输出到当前光标处，再将光标+1

2.3 用 x86 汇编语言实现分支逻辑，循环逻辑和函数调用

- 第一步:



```
linggm@linggm-virtual-machine: ~/lab2/assignment
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
linggm@linggm-virtual-machine:~$ cd lab2/assignment/
linggm@linggm-virtual-machine:~/lab2/assignment$ gedit student.asm

student.asm
~/lab2/assignment 保存(S)

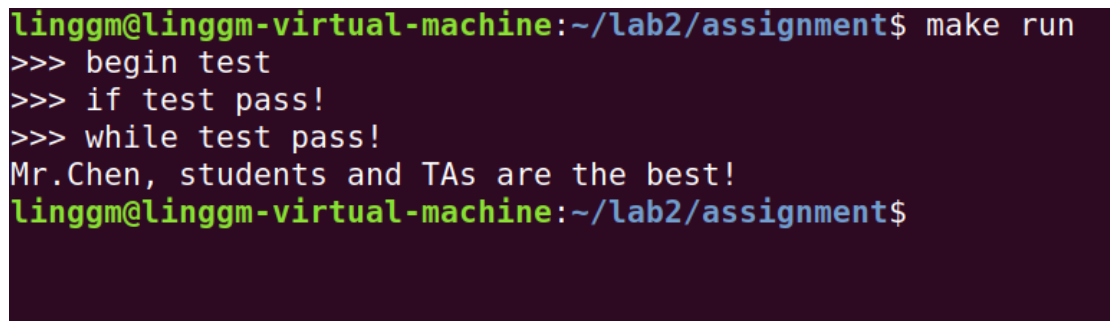
; If you meet compile error, try 'sudo apt install gcc-multilib g++-multilib' first

#include "head.include"
; you code here

your_if:
    cmp byte[a1], 12
    jge branch1 ; else if
    ; if_flag = a1 / 2 + 1
    mov edx, 0 ; 被除数高32位
    mov eax, [a1] ; 被除数低32位
    mov ebx, 2 ; 除数
    idiv ebx ; 除以2, 商在eax中
    inc eax ; +=1 ; 自增
    mov dword[if_flag], eax ; 赋值
    jmp end_if ; 结束

branch1:
    cmp byte[a1], 24
    jge branch2 ; else
    ; if_flag = (24 - a1) * a1
    mov eax, 24
    sub eax, [a1]
    imul eax, [a1]
    mov dword[if_flag], eax
    jmp end_if ; 结束
```

- 第二步



```
linggm@linggm-virtual-machine:~/lab2/assignment$ make run
>>> begin test
>>> if test pass!
>>> while test pass!
Mr.Chen, students and TAs are the best!
linggm@linggm-virtual-machine:~/lab2/assignment$
```

2.4 字符弹射程序

步骤同 2.2，代码见 3.4，结果见 4.4

3 关键代码

3.1 任务 1，输出学号的代码

仅展示与 example1 不同的代码

3.2.1 任务 2，设置光标、获取光标位置的代码

```
    ; 设置光标位置
mov ah, 2    ;功能号
mov bh, 0    ;页码
mov dh, 6    ;行
mov dl, 8    ;列
int 10h     ;调用中断

    ; 初始化d寄存器,设为0
xor dx, dx

    ; 获取光标位置
mov ah, 3    ;功能号
mov bh, 0    ;页码
int 10h     ;调用中断

mov ah, 0x07; 黑底白字

add dh, 48   ;0的ascii码为48
add dl, 48
```

先设置光标位置到第 6 行第 8 列，再初始化 dx 寄存器，然后调用 int10h 中断，将光标位置读入 dx 中，传送到显存中，输出到屏幕上。接下来的部分与 example1 相同，不作展示

3.2.2 任务 2，利用 int10h 中断打印学号的代码

```
; 函数功能：将al处的字符打印在当前光标处
print_al:
    ;要求打印的字符在al处
    mov bl, 0x07 ;黑底白字
    mov bh, 0x00 ;页码为0
    mov cx, 1 ;输出一个字符
    mov ah, 0x09 ;写字符的功能号
    int 10h ;中断
    call cursor_inc
    ret

; 函数功能：当前光标位置+=1
cursor_inc:
    ; 设置光标位置
    mov ah, 2 ;功能号
    mov bh, 0 ;页码
    cmp dl, 79
    jne label ;如果dl的值不等于79，则跳转到label
    add dh, 1 ;如果dl的值等于79，则行号+1
    mov dl, -1 ;如果dl的值等于79，则列号变为-1，经下条指令变为0
label: add dl, 1 ;列坐标+1
    int 10h ;调用中断
    ret
```

函数 print_al 的功能是将 al 处的字符打印在当前光标处，
cursor_inc 的功能是将当前光标位置+1（列坐标在 79 前，列坐标
加一，列坐标在 79 时，列坐标置为 0，行坐标加一），每次更改 al
中字符的值，然后调用 print_al 即可

```
; 打印学号
mov al, '2'
call print_al
mov al, '1'
call print_al
mov al, '3'
call print_al
mov al, '0'
call print_al
mov al, '7'
call print_al
mov al, '0'
call print_al
mov al, '7'
call print_al
mov al, '7'
call print_al
```

3.2.3 任务 2，利用 int16h 中断从键盘获取字符并回显的代码

```
； 函数功能：将a1处的字符打印在当前光标处
print_al:
    ；要求打印的字符在a1处
    mov bl, 0x07    ；黑底白字
    mov bh, 0x00    ；页码为0
    mov cx, 1      ；输出一个字符
    mov ah, 0x09    ；写字符的功能号
    int 10h        ；中断
    call cursor_inc
    ret

； 函数功能：当前光标位置+=1
cursor_inc:
    ； 设置光标位置
    mov ah, 2      ；功能号
    mov bh, 0      ；页码
    cmp dl, 79
    jne label      ；如果dl的值不等于79，则跳转到label
    add dh, 1      ；如果dl的值等于79，则行号+1
    mov dl, -1     ；如果dl的值等于79，则列号变为-1，经下条指令变为0
label: add dl, 1    ；列坐标+1
    int 10h        ；调用中断
    ret
```

接着用上一个任务的两个函数

```
loop: ； 从键盘获取字符并回显
    call read_chr   ； 从键盘获取字符
    call print_al   ； 回显

    jmp loop        ； 死循环

； 函数功能：从键盘获取字符，放到a1处
read_chr:
    mov ah, 0x00    ；功能号，读键盘
    int 16h        ；中断
    ret
```

新增一个函数：利用 int16 中断读取键盘。

然后在一个死循环里不断读键盘，输出，读键盘，输出即可

3.3.1任务 3，分支逻辑

```
your_if:
    cmp byte[a1], 12
    jge branch1 ; else if
    ; if_flag = a1 / 2 + 1
    mov edx, 0 ; 被除数高32位
    mov eax, [a1] ; 被除数低32位
    mov ebx, 2 ; 除数
    idiv ebx ; 除以2, 商在eax中
    inc eax ; +=1 ; 自增
    mov dword[if_flag], eax ; 赋值
    jmp end_if ; 结束

branch1:
    cmp byte[a1], 24
    jge branch2 ; else
    ; if_flag = (24 - a1) * a1
    mov eax, 24
    sub eax, [a1]
    imul eax, [a1]
    mov dword[if_flag], eax
    jmp end_if ; 结束

branch2:
    ; if_flag = a1 << 4
    mov eax, [a1]
    shl eax, 4
    mov dword[if_flag], eax
```

3.3.2任务 3，循环逻辑

```
your_while:
    pushad
    mov ebx, dword[a2]
    mov ecx, dword[while_flag]
loop: cmp ebx, 12
    jl end_while
    push ebx
    push ecx
    call my_random
    pop ecx
    pop ebx
    push ebx
    sub ebx, 12
    ; while_flag[a2 - 12] = eax
    mov [ecx+ebx], al
    pop ebx
    dec ebx
    mov dword[a2], ebx
    jmp loop
end_while: popad
```

3.3.3 任务 3，函数编写

```
your_function:
    pushad
    mov ebx, [your_string]
    mov ecx, 0
    label:
        mov dl, byte[ebx+ecx] |
        cmp dx, 0 ; string[i] != '\0'
        je end_fun
        pushad
        push dx ; 入栈
        call print_a_char
        pop dx
        popad
        inc ecx ; ++i
        jmp label
    end_fun: popad
    ret
```

```
linggm@linggm-virtual-machine:~/lab2/assignment$ gedit student.asm
linggm@linggm-virtual-machine:~/lab2/assignment$ make run
>>> begin test
>>> if test pass!
>>> while test pass!
Mr.Chen, students and TAs are the best!
linggm@linggm-virtual-machine:~/lab2/assignment$
```

3.4 任务 4，字符弹射小程序

```
; 函数功能：取随机数，打印数字的随机数，颜色的随机数
get_random:
    ; 取随机打印数字
    rdtsc ; 获取时钟计数器值，高32位在edx中，低32位在eax中
    mov ecx, 10 ; 设置取模数为10
    idiv ecx ; 除以10取余数，返回随机数是余数，在edx中
    mov ebx, edx ; 在ebx暂存要打印的数字
    add ebx, 48 ; 0的ascii码是48

    ; 取随机打印颜色
    rdtsc ; 获取时钟计数器值，高32位在edx中，低32位在eax中
    mov ecx, 256 ; 设置取模数为256
    idiv ecx ; 除以256取余数，返回随机数是余数，在edx中

    mov eax, ebx
    mov ebx, edx

    ret ;
```

使用随机数，取得打印字符的颜色及数字

```
; 函数功能：睡眠一定时间
sleep:
    mov eax, 1500000 ; 设置需要延迟的时钟周期数
    delay:
    dec eax ; 浪费1个时钟周期
    jnz delay ; 如果还有剩余时钟周期，则继续浪费
    ret ; 返回调用者
```

睡觉

```
; 函数功能：将al处的字符打印在当前光标处
print_al:
    ; 要求打印的字符在al处
    mov bh, 0x00 ; 页码为0
    mov cx, 1 ; 输出一个字符
    mov ah, 0x09 ; 写字符的功能号
    int 10h ; 中断
    call cursor_inc ; 光标+=1
    ret
```

打印一个字符

```

; 函数功能: 当前光标位置+=1
cursor_inc:
    ; 获取光标位置
    mov ah, 3 ;功能号
    mov bh, 0 ;页码
    int 10h ;调用中断

    ; 设置光标位置
    mov ah, 2 ;功能号
    mov bh, 0 ;页码

    branch1: cmp byte[status], 0 ;右下
              jne branch2
              b11: cmp dh, 24 ;行号24
                  jne b12
                  sub dh, 1
                  add dl, 1
                  mov byte[status], 1 ;右上
                  jmp end_1
              b12: cmp dl, 79 ;列号79
                  jne b13
                  sub dl, 1
                  add dh, 1
                  mov byte[status], 3 ;左下
                  jmp end_1
              b13: ;正常
                  add dl, 1
                  add dh, 1
              end_1:
              jmp end_if

```

```

branch4: cmp byte[status], 3 ;左下
          b41: cmp dh, 24 ;行号24
              jne b42
              sub dh, 1
              sub dl, 1
              mov byte[status], 2 ;左上
              jmp end_4
          b42: cmp dl, 0 ;列号0
              jne b43
              add dl, 1
              add dh, 1
              mov byte[status], 0 ;右下
              jmp end_4
          b43: ;正常
              sub dl, 1
              add dh, 1
          end_4:
end_if:
int 10h ;调用中断
ret

```


由于代码过长，仅展示最前面和最后面的部分。主要是嵌套两层的 if-elif-else 语句。外面的一层判断当前移动方向。里面一层判断是否碰到边界，碰到哪个边界，从而决定移动的方向。

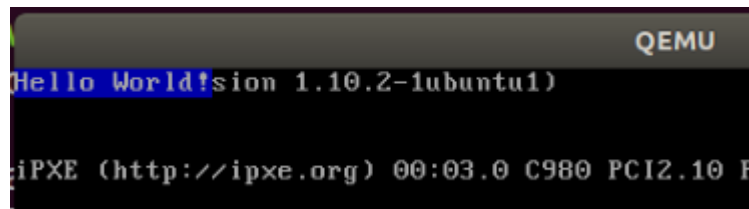
```
loop:
    call get_random
    call print_al
    call sleep
    jmp loop

jmp $ ; 死循环
```

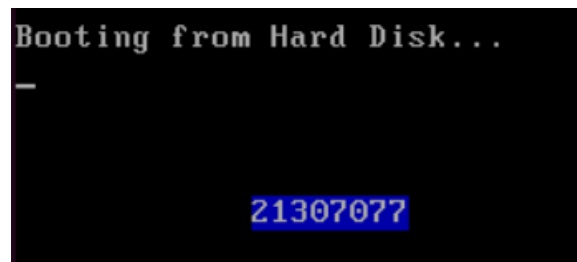
主函数，死循环，每次循环随机决定打印的字符及其颜色，然后将其打印出来，然后光标移动，然后睡眠一段时间，然后继续循环

4 实验结果

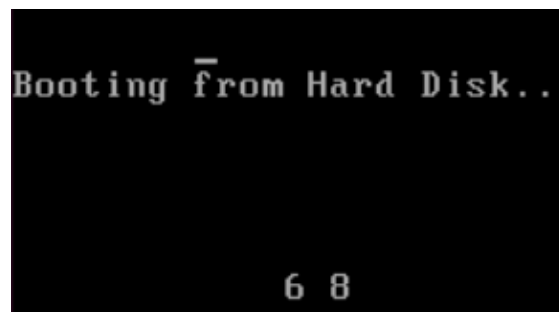
4.1.1 复现 example1



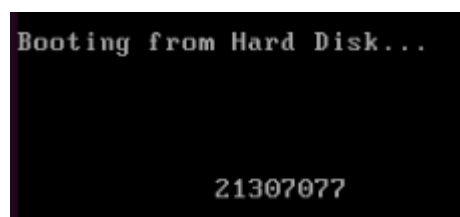
4.1.2 输出学号



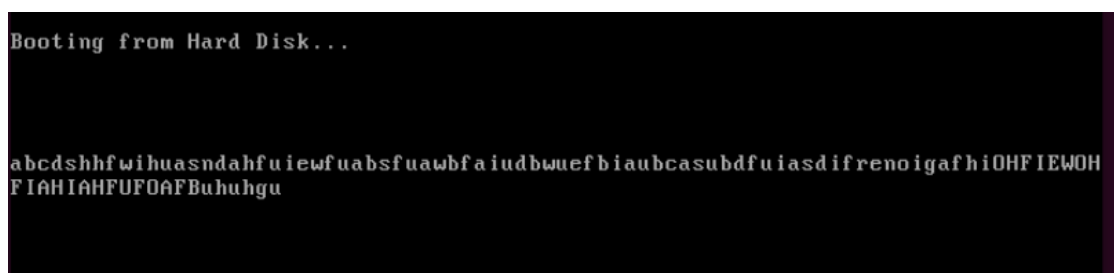
4.2.1 设置光标位置，获取光标位置



4.2.2 利用 int10h 中断输出学号



4.2.3 利用 int16h 从键盘获取字符并回显



4.3

```
linggm@linggm-virtual-machine:~/lab2/assignment$ make run
>>> begin test
>>> if test pass!
>>> while test pass!
Mr.Chen, students and TAs are the best!
linggm@linggm-virtual-machine:~/lab2/assignment$
```

4.4

