

- 1. 实验要求
- 2. 实验步骤/实验过程/关键操作/关键代码/实验结果（图文结合，必要时提供流程图,提供关键操作/关键代码的解释、原理等,截图，并对结果解释）
  - 编写MBR
  - debug
  - 思考题
    - Q16
      - (1)
      - (2)
      - (3)
    - Q17
      - (1)
      - (2)
      - (3)
    - Q14
- 3. 总结(对实验过程中遇到的问题进行总结，可以提出对实验设置的改进意见)



实验课程: 操作系统

实验名称: 编译内核/利用已有内核构建OS

专业名称: 计算机科学与技术

学生姓名: 吴臻

学生学号: 21307371

实验地点: 实验中心大楼D栋501

实验成绩:

报告时间: 2023/3/17

## 1. 实验要求

---

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,能自己编写程序, 然后让计算机在启动后加载运行,并学习如何使用gdb来调试程序的基本方法。

## 2. 实验步骤/实验过程/关键操作/关键代码/实验结果（图文结合，必要时提供流程图，提供关键操作/关键代码的解释、原理等,截图，并对结果解释）

---

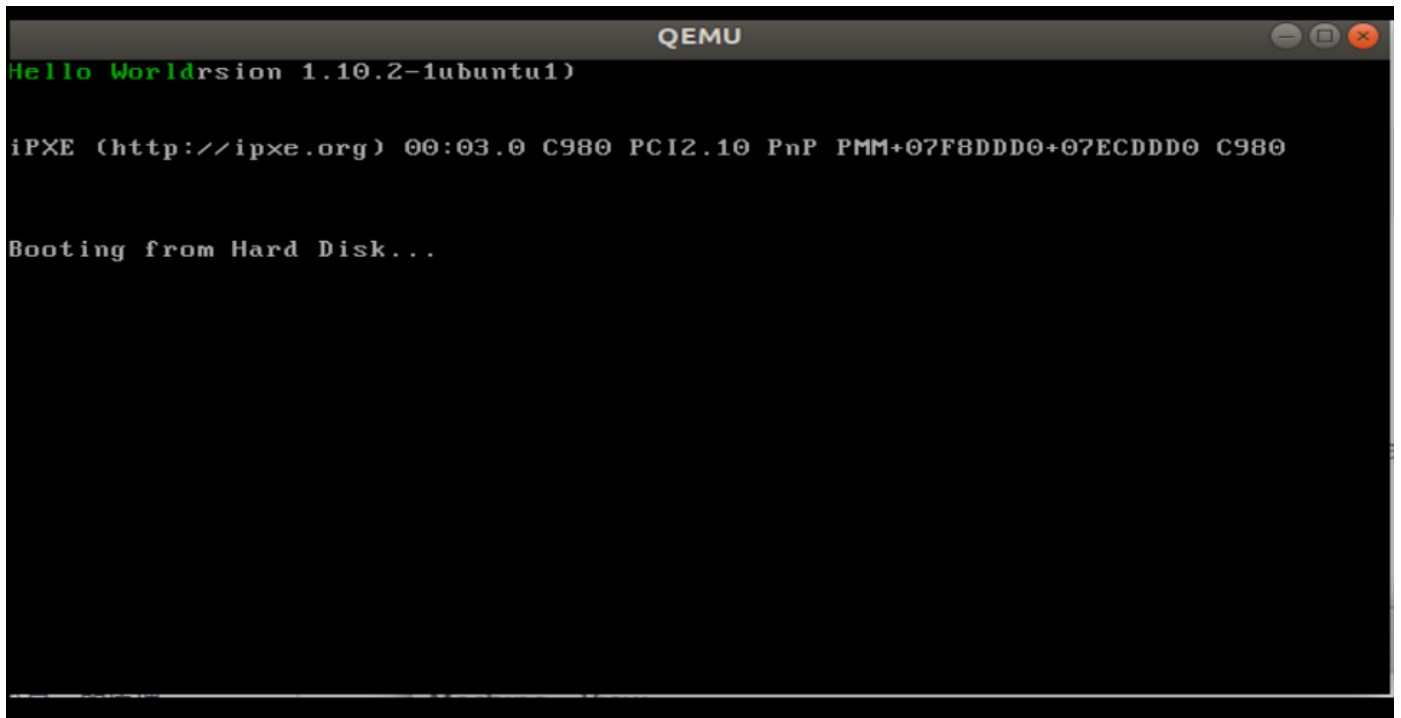
### 编写MBR

---

```
; 初始化栈指针
mov sp, 0x7c00
mov ax, 0xb800 ;显存位置
mov gs, ax
;ax的高八位表示颜色，低八位表示字符
mov ah, 0x01 ; 青色
mov al, 'H'
mov [gs:2 * 0], ax
```

编写完写入img并运行

```
nasm -f bin mbr.asm -o mbr.bin
qemu-img create hd.img 10m
dd if=mbr.bin of=hd.img bs=512 count=1 seek=0 conv=notrunc
qemu-system-i386 -hda hd.img -serial null -parallel stdio
```



## debug

注意，生成符号表的时候需要将 `mbr.asm` 开头的 `org` 伪指令删去。

```
nasm -o mbr.o -g -f elf32 mbr.asm
ld -o mbr.symbol -melf_i386 -N mbr.o -Ttext 0x7c00
```

为了使 `qemu` 支持 `gdb` 来进行 `debug`，我们需要向 `qemu` 的启动命令中加入 `-s -S` 参数，如下所示。

```
qemu-system-i386 -hda hd.img -s -S -parallel stdio -serial null
```

然后在另外一个 `Terminal` 进入 `gdb`。

```
gdb
```

使用 `gdb` 连上 `qemu` 即可，注意以下命令是在进入 `gdb` 后输入的。

```
target remote:1234
```

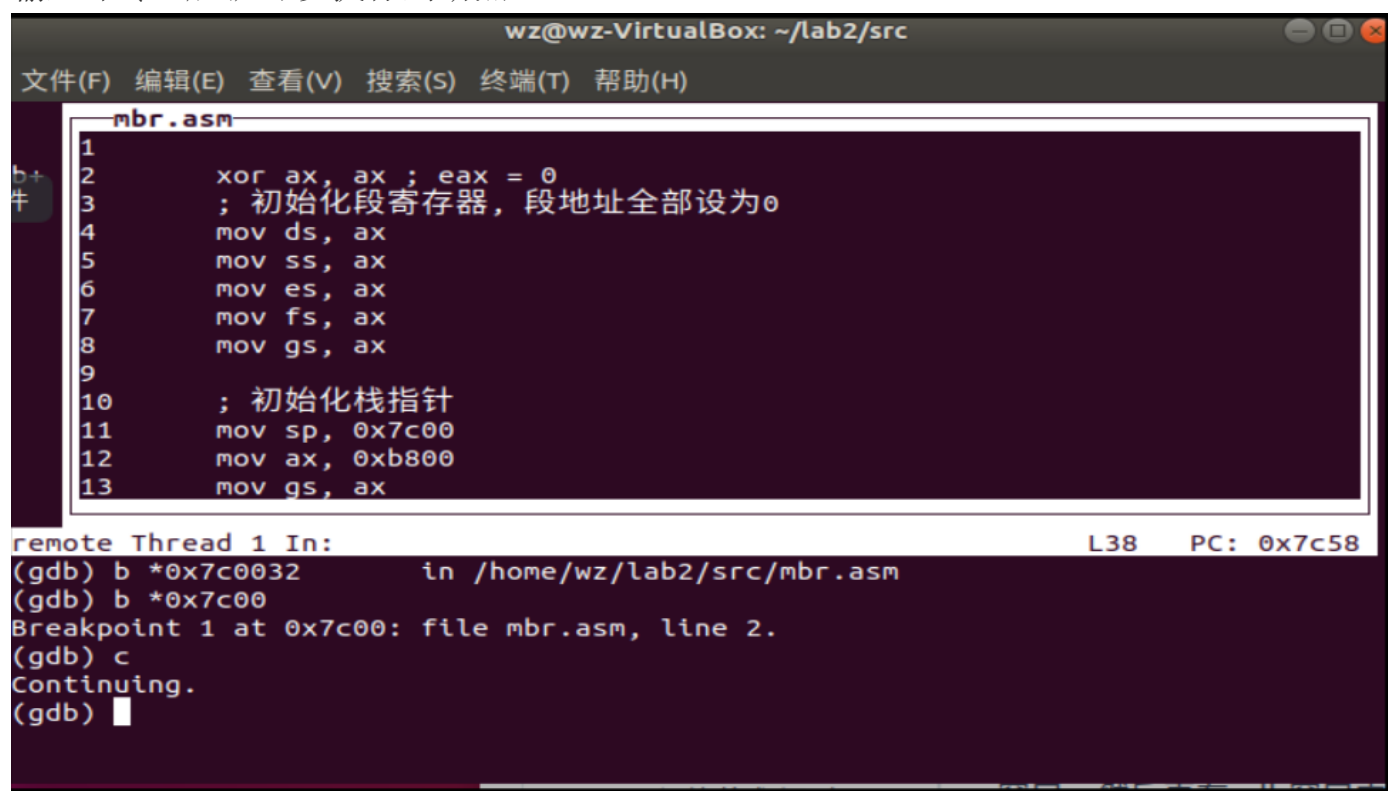
我们加载符号表，打开代码显示窗口。

```
add-symbol-file mbr.symbol 0x7c00
layout src
```

接着，在 `0x7c00`处设置断点。

```
b *0x7c00
```

输入命令 `c`后就可以执行到断点



```
wz@wz-VirtualBox: ~/lab2/src
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
mbr.asm
1
2     xor ax, ax ; eax = 0
3     ; 初始化段寄存器，段地址全部设为0
4     mov ds, ax
5     mov ss, ax
6     mov es, ax
7     mov fs, ax
8     mov gs, ax
9
10    ; 初始化栈指针
11    mov sp, 0x7c00
12    mov ax, 0xb800
13    mov gs, ax

remote Thread 1 In:                                     L38    PC: 0x7c58
(gdb) b *0x7c0032          in /home/wz/lab2/src/mbr.asm
(gdb) b *0x7c00
Breakpoint 1 at 0x7c00: file mbr.asm, line 2.
(gdb) c
Continuing.
(gdb) █
```

## 思考题

### Q16

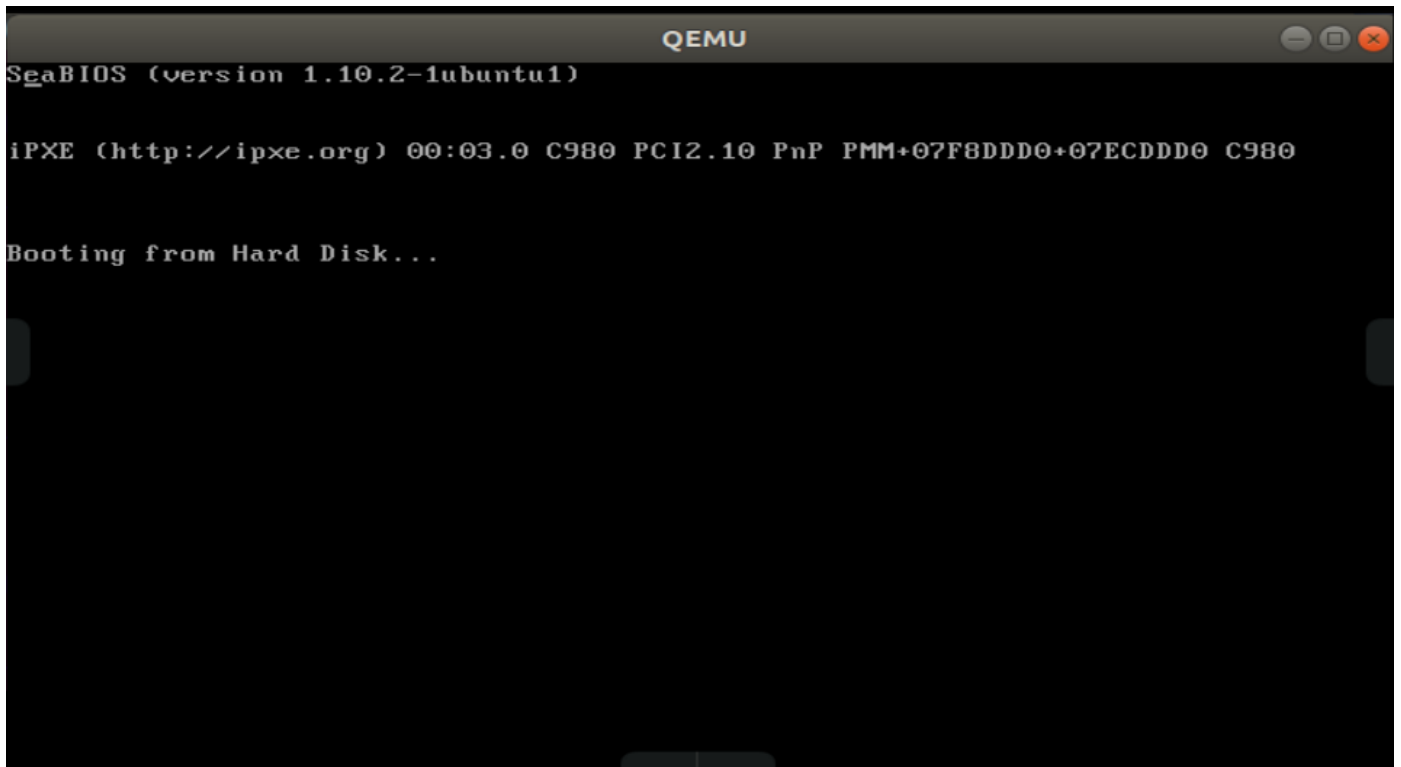
(1)

```
mov ah, 0x03      ; 设置BIOS中断功能号为0x03，表示获取光标位置
mov bh, 0x00      ; 设置显示页面号为0
int 0x10          ; 调用BIOS中断0x10
mov ah, 0x02      ; 设置BIOS中断功能号为0x02，表示设置光标位置
mov bh, 0x00      ; 设置显示页面号为0
```

```

mov dh, dl      ; 将光标的列号移动到行号中
mov dl, 0x01    ; 设置光标所在的列号
int 0x10        ; 调用BIOS中断0x10

```



## (2)

```

msg db "21307371", 0    ; 字符串末尾需要添加一个null字符

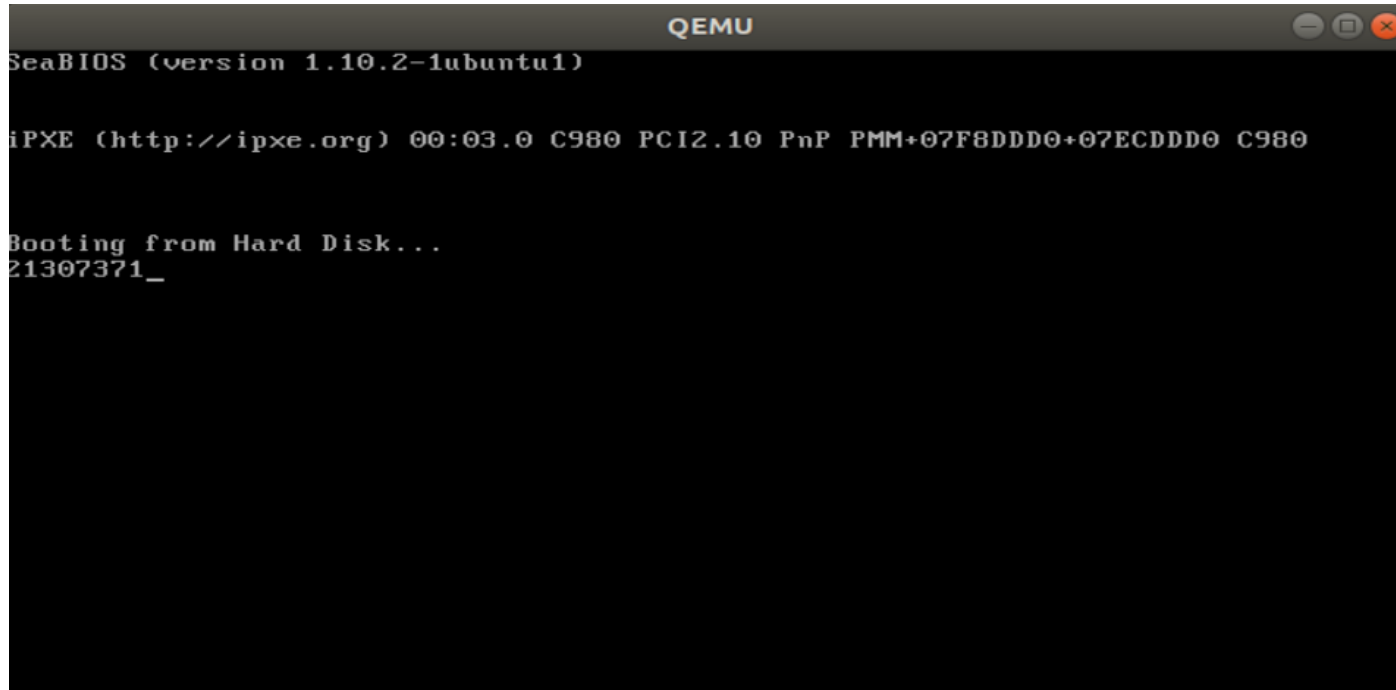
start:
    mov ax, 0            ; 清空AX寄存器
    mov ds, ax           ; 将DS寄存器设置为0
    mov ah, 0x0e
    mov bh, 0

    mov si, msg           ; 将SI寄存器设置为字符串的地址
    call print_str        ; 调用打印字符串的函数

    ; 无限循环
    jmp $                 ; 跳回当前地址

print_str:
    lodsb                ; 从SI指向的地址读取一个字节到AL寄存器, 并将SI自动加1
    or al, al             ; 检查AL寄存器是否为0
    jz done               ; 如果为0, 结束函数
    int 0x10              ; 调用屏幕中断服务程序, 打印字符
    jmp print_str         ; 继续打印下一个字符
done:
    ret                  ; 返回

```

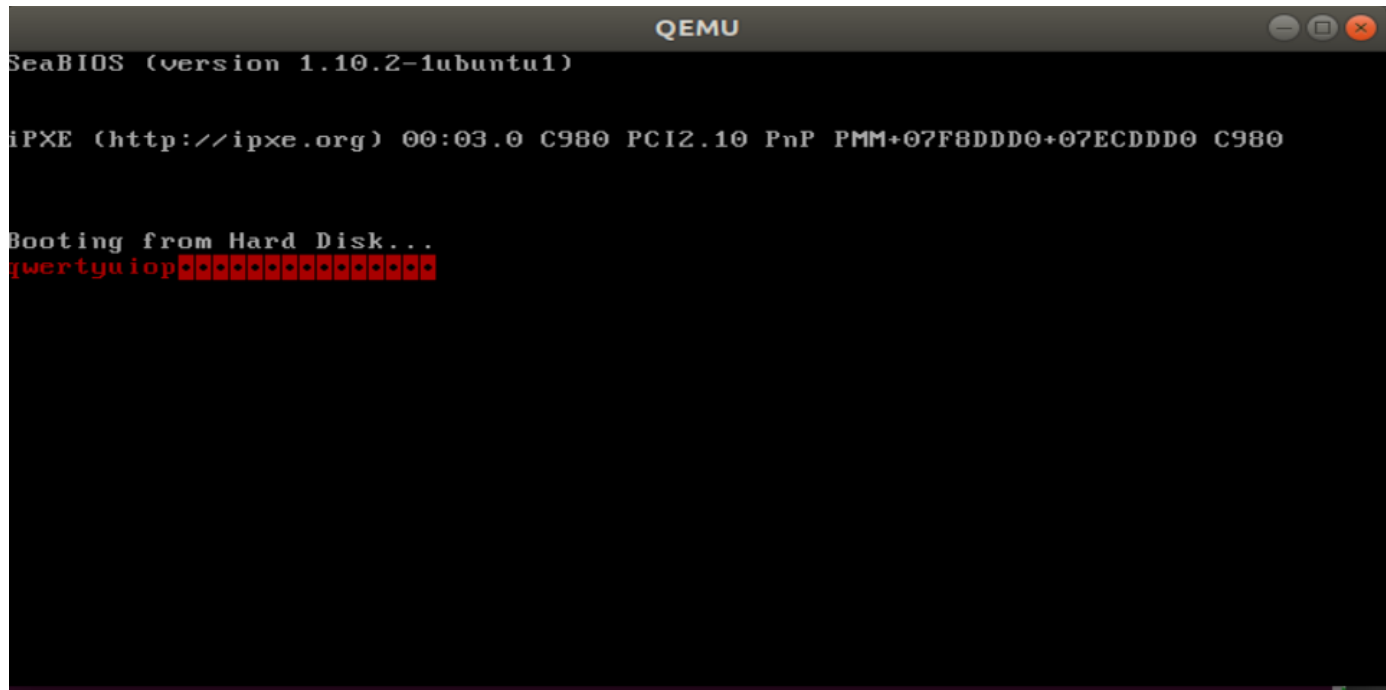


(3)

```
keyboard_input:
mov ah, 0x00      ;键盘输入
int 0x16
cmp al,0x1b      ;判断输入是否为esc, 若是则结束
je end

input:
;设置光标位置
mov ah ,0x02
mov bh, 0x00
int 0x10
add dl, 1        ;移向下一位
mov ah, 0x09
mov bh ,0x00
mov bl,0x04
mov cx,0x01
int 0x10        ;打印字符

jmp keyboard_input
```



## Q17

注意变量名要加[]

(1)

```
your_if:
mov eax,[a1]
xor ebx,ebx
cmp eax, 12      ;比较a1和12的大小
jl if_less_than_12 ;如果a1小于12, 跳转到if_less_than_12标签
cmp eax, 24      ;比较a1和24的大小
jl if_less_than_24 ;如果a1小于24, 跳转到if_less_than_24标签
shl eax, 4       ;将a1左移4位, 相当于a1乘以16
mov ebx, eax     ;将结果存储到if_flag中
jmp end_if       ;跳转到end_if标签, 结束if语句
if_less_than_12:
shr eax, 1       ;将a1右移1位, 相当于除以2
add eax, 1       ;将a1除以2后再加1
mov ebx, eax     ;将结果存储到if_flag中
jmp end_if       ;跳转到end_if标签, 结束if语句
if_less_than_24:
mov ebx, 24      ;将24存储到if_flag中
sub ebx, eax     ;计算24-a1的值
imul ebx, eax    ;将24-a1乘以a1
end_if:
mov [if_flag],ebx
```

(2)

```

your_while:
mov ecx, [a2]           ; 将a2存储到ECX寄存器中
while_loop:
    cmp ecx, 12         ; 比较ECX和12
    jl end_while       ; 如果ECX < 12, 则跳出while循环
    call my_random      ; 调用my_random函数, 将随机数存储在EAX中
    mov edx,ecx
    sub edx,12
    add edx,[while_flag]
    mov [edx], eax      ; 将随机数存储在while_flag数组中
    dec ecx             ; 将ECX减1
    jmp while_loop      ; 跳转到while循环的开头
end_while:
mov [a2],ecx

```

### (3)

```

your_function:
; put your implementation here
mov ecx,[your_string]   ;将your_string存入ecx寄存器
my_while:
    pushad              ;将寄存器推入堆栈
    push word [ecx];
    mov di, [ecx]
    call print_a_char
    pop bx
    popad
    inc ecx              ;ecx递增指向下一个字符
    mov al,[ecx]
    cmp al,0
    jne my_while

ret

```



```
wz@wz-VirtualBox: ~/lab2/Q17
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
wz@wz-VirtualBox:~/lab2/Q17$ make run
>>> begin test
>>> if test pass!
>>> while test pass!
Mr.Chen, students and TAs are the best!
wz@wz-VirtualBox:~/lab2/Q17$
```

## Q14

```
;清屏
mov ah,0x00
mov ax,3
int 0x10
;移动光标
mov ah,0x02
mov bh,0
mov dh,0
mov dl,34
int 0x10
```

打印名字和学号与Q16类似

```
;字符弹射

drow dw 1          ;每次row的位移
dcol dw 1          ;每次col的位移
;初始位置
row dw 2
col dw 0

mov al,0x30        ;al初始为字符0
Start:
    push ax
    mov dx,[row]    ;dx--row
    mov cx,[col]    ;cx--col
    Row_judge:
```

```

        cmp dx,1
        je down
        cmp dx,23
        je up
Col_judge:
        cmp cx,0
        je right
        cmp cx,79
        je left
;未触及边界
        jmp my_print

up:
        mov ax,-1
        mov [drow],ax
        jmp Col_judge
down:
        mov ax,1
        mov [drow],ax
        jmp Col_judge
left:
        mov ax,-1
        mov [dcol],ax
        jmp my_print
right:
        mov ax,1
        mov [dcol],ax
        jmp my_print

my_print:
        pop ax
        ;得到当前位置
        add dx,[drow]
        add cx,[dcol]
        mov [row],dx
        mov [col],cx
        ;设置数字
        mov ah,dl
        add al,3
        cmp al,0x3a
        jl less_then
        sub al,10
less_then:
        ;打印
        imul bx,dx,80
        add bx,cx
        imul bx,2
        mov [gs:bx],ax
        ;对称打印
        sub al,1
        ;计算对称Row
        mov bx, 24
        sub bx, dx
        mov dx, bx
        ;计算对称Col
        mov bx, 80
        sub bx, cx

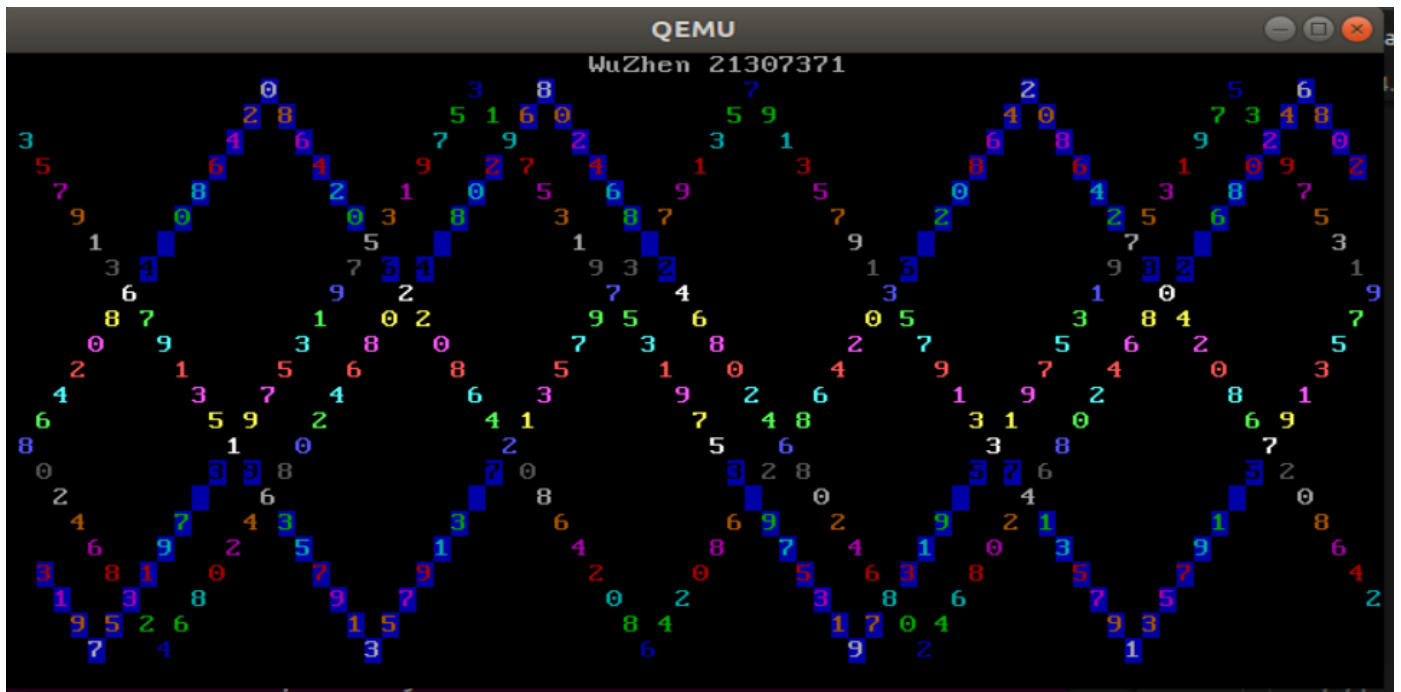
```

```

mov cx, bx
;打印
imul bx, dx, 80
add bx, cx ; cor = 80 * row + col
imul bx, 2
mov [gs:bx], ax

;延迟模块
push cx
mov cx, 0x7777
delay_1:
push cx
mov cx, 0x77
delay_2:
loop delay_2
pop cx
loop delay_1
pop cx
;-----
jmp Start

```



### 3. 总结(对实验过程中遇到的问题进行总结，可以提出对实验设置的改进意见)

汇编代码的编写比起高级语言有着一定的不同，虽然大致逻辑清楚，但是有很多细节需要注意，寄存器的选择也不能随意选择 改进：可以简化代码

```

nasm -f bin mbr.asm -o mbr.bin
qemu-img create hd.img 10m

```

```
dd if=mbr.bin of=hd.img bs=512 count=1 seek=0 conv=notrunc  
qemu-system-i386 -hda hd.img -serial null -parallel stdio
```

```
NAME=Q14  
nasm -f bin $NAME.asm -o $NAME.bin  
dd if=$NAME.bin of=hd.img bs=512 count=1 seek=0 conv=notrunc  
qemu-system-i386 -hda hd.img -serial null -parallel stdio
```