



中山大學
SUN YAT-SEN UNIVERSITY

本科生实验报告

实验课程: 操作系统原理实验

任课教师: 刘宁

实验题目: 第二章 实验入门

专业名称: 信息与计算科学

学生姓名: 郑鸿鑫

学生学号: 22336313

实验地点: 实验中心 D503

实验时间: 2024/3/11

Section 1 实验概述

本次实验中，主要目的是学习 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。根据所学的知识，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。同时，也将学习如何使用 gdb 来调试程序。

Section 2 预备知识与实验环境

- 预备知识：x86 汇编语言程序设计、IA-32 处理器体系结构，字符显示的原理
- 实验环境：
 - 虚拟机版本/处理器型号：
11th Gen Intel® Core™ i5-11320H @ 3.20GHz × 2
 - 代码编辑环境：VS Code
 - 代码编译工具：g++
 - 重要三方库信息：Linux 内核版本号：linux-5.10.210
Ubuntu 版本号：Ubuntu 18.04.6LTS，Busybox 版本号：
Busybox_1_33_0

Section 3 实验任务

- 实验任务 1：学习计算机开机启动过程，启动操作系统，输出 Hello world
- 实验任务 2：学习汇编程序设计与实模式中断，完成课后思考题 16
- 实验任务 3：学习汇编程序设计与字符显示原理，完成课后思考题 17
- 实验任务 4：学习汇编程序设计与字符显示原理，写一个字符弹射程序，完成课后思考题 14

Section 4 实验步骤与实验结果

----- 实验任务 1 -----

- 任务要求：学习计算机开机启动过程，启动操作系统，输出 Hello world
- 思路分析：根据指导书的介绍，了解计算机启动操作系统的过程。
- 实验步骤：

1. 了解计算机开机启动过程：

主要分为加电开机，BIOS 启动，加载 MBR，硬盘启动，内核启动几个步骤。

2. 学习字符显示原理，编写 mbr.asm

代码如下：

```
;org 0x7c00
[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, 0x03 ;青色
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, ' '
mov [gs:2 * 5], ax
mov al, 'W'
mov [gs:2 * 6], ax
mov al, 'o'
mov [gs:2 * 7], ax
mov al, 'r'
mov [gs:2 * 8], ax
mov al, 'l'
mov [gs:2 * 9], ax
mov al, 'd'
mov [gs:2 * 10], ax
jmp $ ; 死循环
times 510 - ($ - $$) db 0
db 0x55, 0xaa
```

3. 对代码进行编译和加载运行。

用 nasm 汇编器编译成二进制文件；

```
nasm -f bin mbr.asm -o mbr.bin
```

生成了 MBR 后，我们将其写入到硬盘的首扇区。我们首先创建一个“硬盘”，这个“硬盘”并不是一个真实的硬盘，实际上是一个预先指定大小的文件而已，又被称为“虚拟磁盘”。硬盘的创建使用的是 qemu-img：

```
qemu-img create hd.img 10m
```

然后将 MBR 写入 hd.img 的首扇区，写入的命令使用的是 linux 下的 dd 命令

```
dd if=mbr.bin of=hd.img bs=512 count=1 seek=0 conv=notrunc
```

参数的解释如下。

- if 表示输入文件。
- of 表示输出文件。
- bs 表示块大小，以字节表示。
- count 表示写入的块数目。
- seek 表示越过输出文件中多少块之后再写入。
- conv=notrunc 表示不截断输出文件，如果不加上这个参数，那么硬盘在写入后多余部份会被截断。

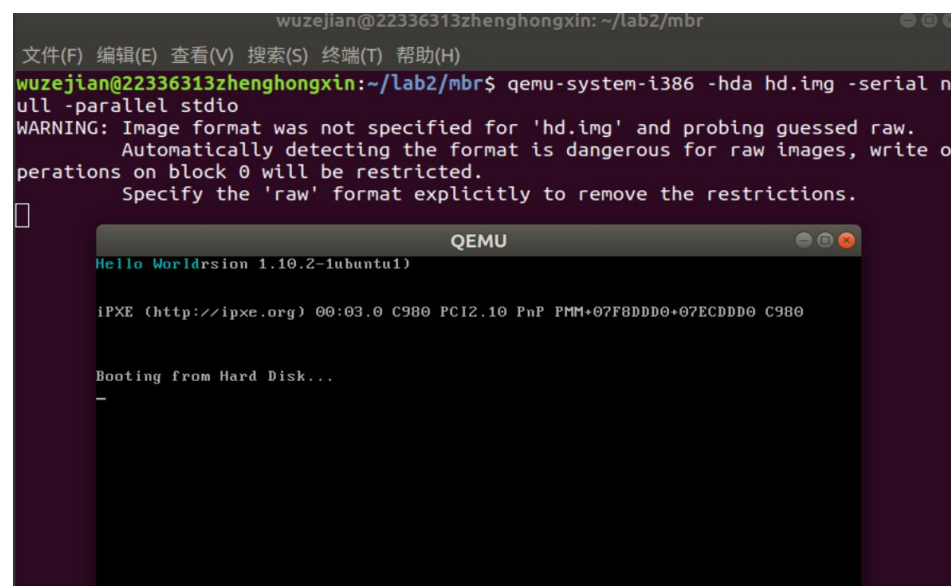
写入 MBR 后我们就可以启动 qemu 来模拟计算机启动了，命令如下：

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

- -hda hd.img 表示将文件 hd.img 作为第 0 号磁盘映像。
- -serial dev 表示重定向虚拟串口到空设备中。
- -parallel stdio 表示重定向虚拟并口到主机标准输入输出设备中。

● 实验结果展示：

启动后的效果如下。可以看到第一行已经输出“Hello World”：



----- 实验任务 2 -----

- 任务要求：学习汇编程序设计 with 实模式中断，完成课后思考题 16
- 思路分析：依照指导书的指引学习实模式中断，参考其他资料学习利用中断对光标进行操作。资料如下：

- [OSDev 关于 BIOS 的介绍](#)
- [BIOS 中断表](#)
- [VIDEO - WRITE CHARACTER ONLY AT CURSOR POSITION](#)
- [VIDEO - WRITE CHARACTER AND ATTRIBUTE AT CURSOR POSITION](#)
- [VIDEO - WRITE STRING \(AT and later, EGA\)](#)
- [VIDEO - GET CURSOR POSITION AND SIZE](#)
- [10h 中断](#)

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

● 实验步骤：

一．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。

编写汇编代码如下：

实现将光标位置设置在（10，10），并且在当前位置输出字符‘n’。

```
;org 0x7c00
[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, 2
mov bh, 0
mov dh, 10
mov dl, 10
```

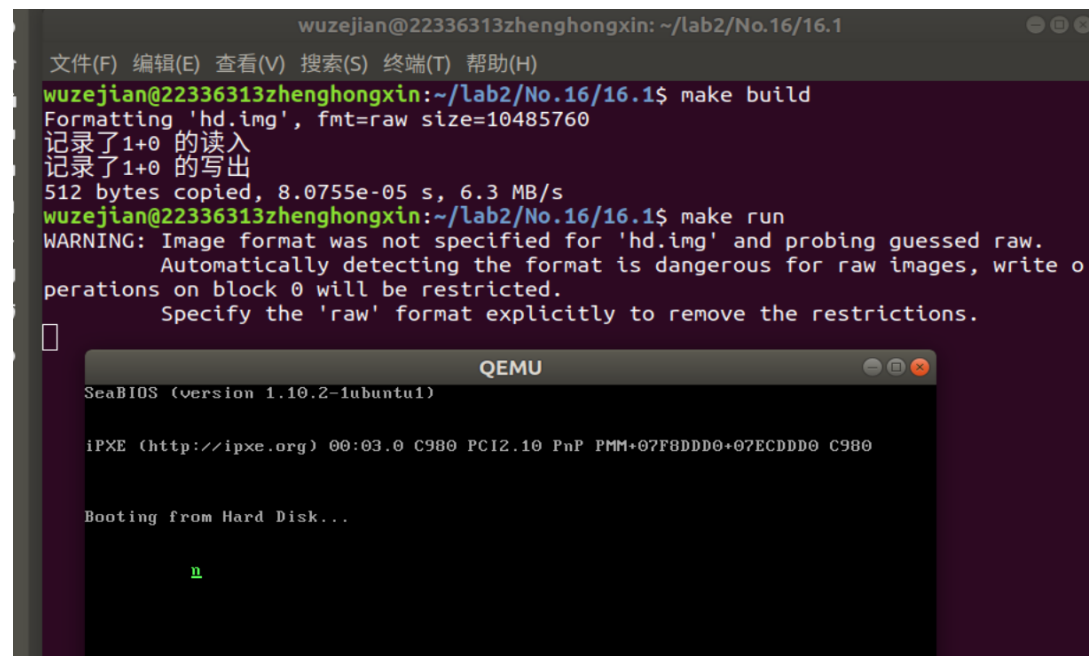
```

int 10h
mov ah, 9
mov al, 'n'
mov bl, 0x0a
mov cx, 1
int 10h
jmp $ ; 死循环
times 510 - ($ - $$) db 0
db 0x55, 0xaa

```

对代码进行编译和加载运行。由于命令行输入命令的方式过于繁琐，为了方便后续其他代码的编译运行，本人参考了参考书，学习了如何编写和使用 makefile 文件，可以利用 make build 进行编译，make run 进行运行，结果如下：

(makefile 代码详见下文 Section 6)



```

wuzejian@22336313zhenghongxin: ~/lab2/No.16/16.1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
wuzejian@22336313zhenghongxin:~/lab2/No.16/16.1$ make build
Formatting 'hd.img', fmt=raw size=10485760
记录了1+0 的读入
记录了1+0 的写出
512 bytes copied, 8.0755e-05 s, 6.3 MB/s
wuzejian@22336313zhenghongxin:~/lab2/No.16/16.1$ make run
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 PMM+07F8DDDD+07ECDDDD C980

Booting from Hard Disk...

u

```

二. 修改 Hello World 的代码，使用实模式下的中断来输出学号：

编写汇编代码如下：

打印学号 22336313

```

;org 0x7c00
[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

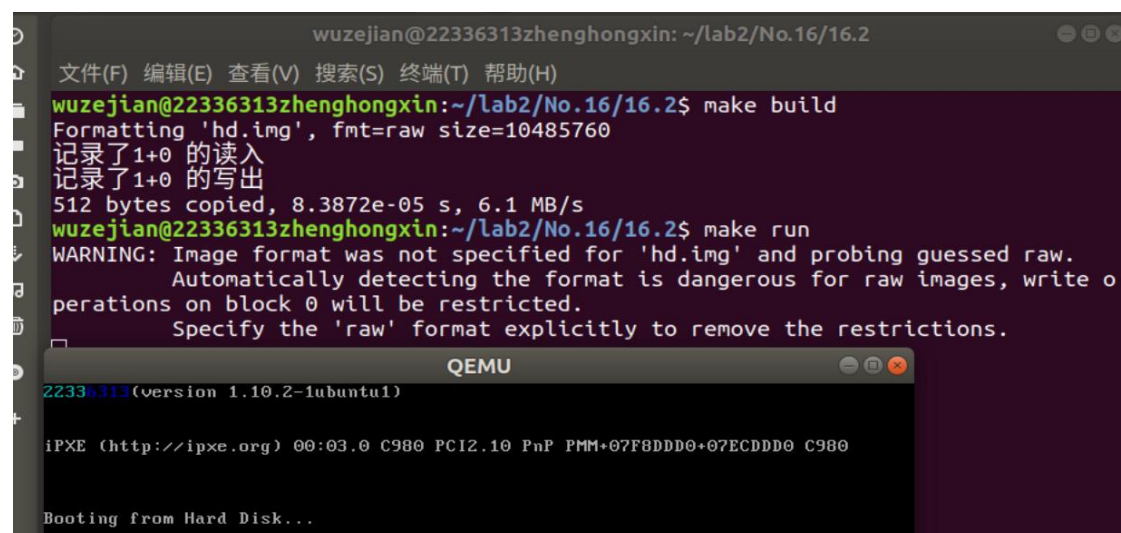
```

```

mov gs, ax
mov ah, 0x03 ;
mov al, '2'
mov [gs:2 * 0], ax
mov al, '2'
mov [gs:2 * 1], a
mov al, '3'
mov [gs:2 * 2], ax
mov al, '3'
mov [gs:2 * 3], ax
mov ah, 0x01
mov al, '6'
mov [gs:2 * 4], ax
mov al, '3'
mov [gs:2 * 5], ax
mov al, '1'
mov [gs:2 * 6], ax
mov al, '3'
mov [gs:2 * 7], ax
jmp $ ; 死循环
times 510 - ($ - $$) db 0
db 0x55, 0xaa

```

同样利用 makefile 文件方便进行操作，通过编译运行后结果如下：



```

wuzejian@22336313zhenghongxin: ~/lab2/No.16/16.2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
wuzejian@22336313zhenghongxin:~/lab2/No.16/16.2$ make build
Formatting 'hd.img', fmt=raw size=10485760
记录了1+0 的读入
记录了1+0 的写出
512 bytes copied, 8.3872e-05 s, 6.1 MB/s
wuzejian@22336313zhenghongxin:~/lab2/No.16/16.2$ make run
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
22336313 (version 1.10.2-1ubuntu1)
iPXE (http://ipxe.org) 00:03:0 C980 PCI2.10 PnP PMM+07F8DDDD+07ECDDDD C980
Booting from Hard Disk...

```

三. 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

编写汇编代码如下：

实现键盘输入并回显

```

;org 0x7c00
[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

```

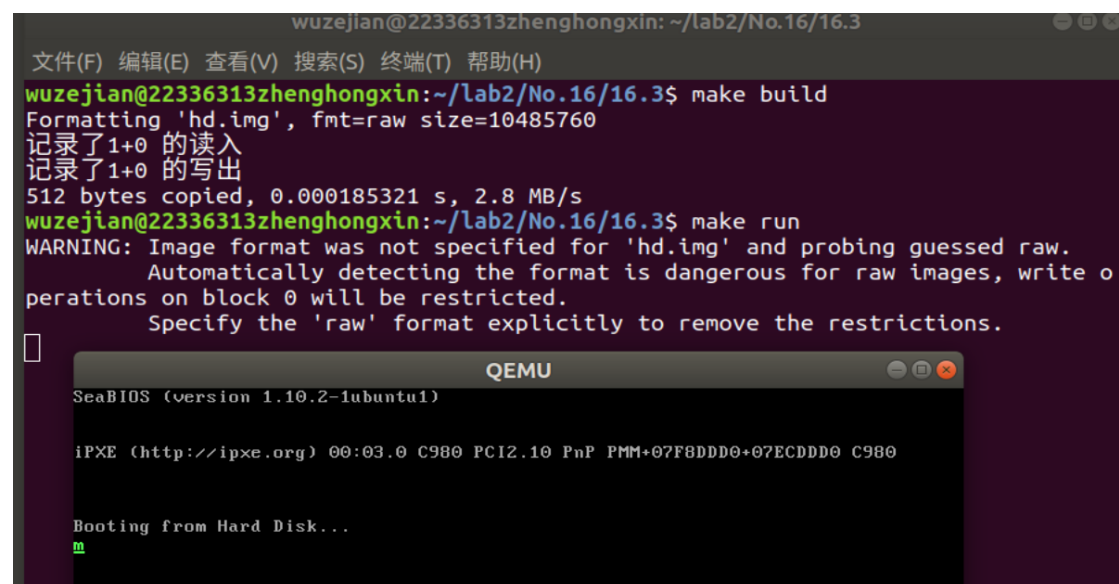
```

; 初始化栈指针
mov sp, 0x7c00
mov ax, 0xb800
mov gs, ax
mov ah, 0
int 16h
cmp al, 1bh
mov ah, 9
mov bh, 0
mov bl, 0x0a
mov cx, 1
int 10h
jmp $ ; 死循环
times 510 - ($ - $$) db 0
db 0x55, 0xaa

```

同样利用 makefile 文件方便进行操作，通过编译运行后结果如下：

（键盘键入 ‘m’，终端成功回显）



```

wuzejian@22336313zhenghongxin: ~/lab2/No.16/16.3
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
wuzejian@22336313zhenghongxin:~/lab2/No.16/16.3$ make build
Formatting 'hd.img', fmt=raw size=10485760
记录了1+0 的读入
记录了1+0 的写出
512 bytes copied, 0.000185321 s, 2.8 MB/s
wuzejian@22336313zhenghongxin:~/lab2/No.16/16.3$ make run
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.

```

----- 实验任务 3 -----

- 任务要求：学习汇编程序设计，完成课后思考题 17
- 思路分析：依照指导书指引，学习汇编语言基础。实现提示如下：
 - 寄存器请使用 32 位的寄存器。
 - 首先执行命令 `sudo apt install gcc-multilib g++-multilib` 安装相应环境。
 - 你需要实现的代码文件在 `assignment/student.asm` 中。
 - 编写好代码之后，在目录 `assignment` 下使用命令 `make run` 即可测试，不需要放到 `mbr` 中使用 `qemu` 启动。

- `a1`、`if_flag`、`my_random` 等都是预先定义好的变量和函数，直接使用即可。
- 你可以修改 `test.cpp` 中的 `student_setting` 中的语句来得到你想要的 `a1,a2`。

● 实验步骤：

一．分支逻辑的实现。将下列伪代码转换成汇编代码，并放置在标号 `your_if` 之后。

```
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
```

二．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
```

三．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数组 `string`。

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

根据上述要求编写汇编代码如下：

```
%include "head.include"
; you code here
your_if:
    mov eax, [a1]
    cmp eax, 12
    jl less_than_12
    cmp eax, 24
    jl less_than_24
    shl eax, 4
    mov [if_flag], eax
    jmp end_if
less_than_12:
    mov eax, [a1]
    shr eax, 1
```

```

    inc eax
    mov [if_flag], eax
    jmp end_if
less_than_24:
    mov eax, 24
    sub eax, [a1]
    imul eax, [a1]
    mov [if_flag], eax
end_if:
    ; Your code after the if statement
your_while:
    mov ecx, [a2]                ;Load a2 into ecx
while_condition:
    cmp ecx, 12                  ;Compare ecx with 12
    jl end_while                 ;Jump to end_while if ecx < 12
    call my_random               ;Call my_random function
    mov [while_flag + ecx - 12], al
    dec ecx                      ; Update a2
    jmp while_condition
    ; Store the random number in while_flag array
end_while:
    mov [a2], ecx
%include "end.include"
your_function:
    mov ecx, 0
    mov ebx, [your_string]
for_condition:
    mov edx, ecx
    add edx, ebx
    mov al, [edx]
    cmp al, 0
    je end_function
    pushad
    push ax
    call print_a_char
    pop ax
    popad
    inc ecx
    jmp for_condition
end_function:
    ret

```

- 实验结果展示：通过执行前述代码，可得下图结果：

```

wuzajian@22336313zhenghongxin: ~/lab2/No.17/assignment
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
wuzajian@22336313zhenghongxin:~/lab2/No.17/assignment$ make run
>>> begin test
>>> if test pass!
>>> while test pass!
Mr.Chen, students and TAs are the best!
wuzajian@22336313zhenghongxin:~/lab2/No.17/assignment$

```

----- 实验任务 4 -----

- 任务要求：学习汇编程序设计 with 字符显示原理，写一个字符弹射程序，完成课后思考题 14

- 思路分析：依照指导书的指引，学习字符显示原理，完成实验。
- 实验步骤：

按照实验要求编写汇编代码如下：

大致思路为先清屏，然后设置起点坐标，然后每次打印字符后将显示字符的坐标更新，在碰到边界时需要修改运动的方向。

```
[bits 16]
;org 0x7c00
xor ax, ax
mov ds, ax
mov ss, ax
mov es, ax
mov fs, ax
mov gs, ax
mov sp, 0x7c00
mov ax, 0xb800
mov gs, ax
mov ah, 0x6
mov bh, 0x7
mov cx, 0
mov dx, 0x184f
int 10h
mov eax, 2
mov ebx, 0
mov ecx, 0 ;用于记录当前弹射状态
mov si, 1
location:
    push eax
    push ebx
    push ecx ;进栈保存现场
    mov cx, 80
    mul cx ; eax = eax * 80
    add eax, ebx ; eax = eax + ebx
    shl eax, 1 ; eax = eax * 2
    mov di, ax ; 将计算结果保存到 di 寄存器
    mov ah, [si]
    inc si
    mov al, 'o' ;打印字符
    mov [gs:di], ax
    mov ah, 0x86
    mov al, 0
    mov cx, 0x1
    mov dx, 0x0
    int 15h
    pop ecx ;恢复现场
    pop ebx
    pop eax
    cmp ecx, 0
    je d_r
    cmp ecx, 1
    je u_r
    cmp ecx, 2
    je d_l
    cmp ecx, 3
```

```

        je u_l
d_r:
    mov ecx, 0
    cmp al, 24
    je u_r ;从右下转右上
    cmp bl, 79
    je d_l ;从右下转左下
    inc eax
    inc ebx
    jmp location
u_r:
    mov ecx, 1
    cmp al, 0
    je d_r ;从右上转右下
    cmp bl, 79
    je u_l ;从右下转左上
    dec eax
    inc ebx
    jmp location
d_l:
    mov ecx, 2
    cmp al, 24
    je u_l ;从左下转左上
    cmp bl, 0
    je d_r ;从左下转右下
    inc eax
    dec ebx
    jmp location
u_l:
    mov ecx, 3
    cmp al, 0
    je d_l ;从左上转左下
    cmp bl, 0
    je u_r ;从左上转右上
    dec eax
    dec ebx
    jmp location
end:
    jmp $

```

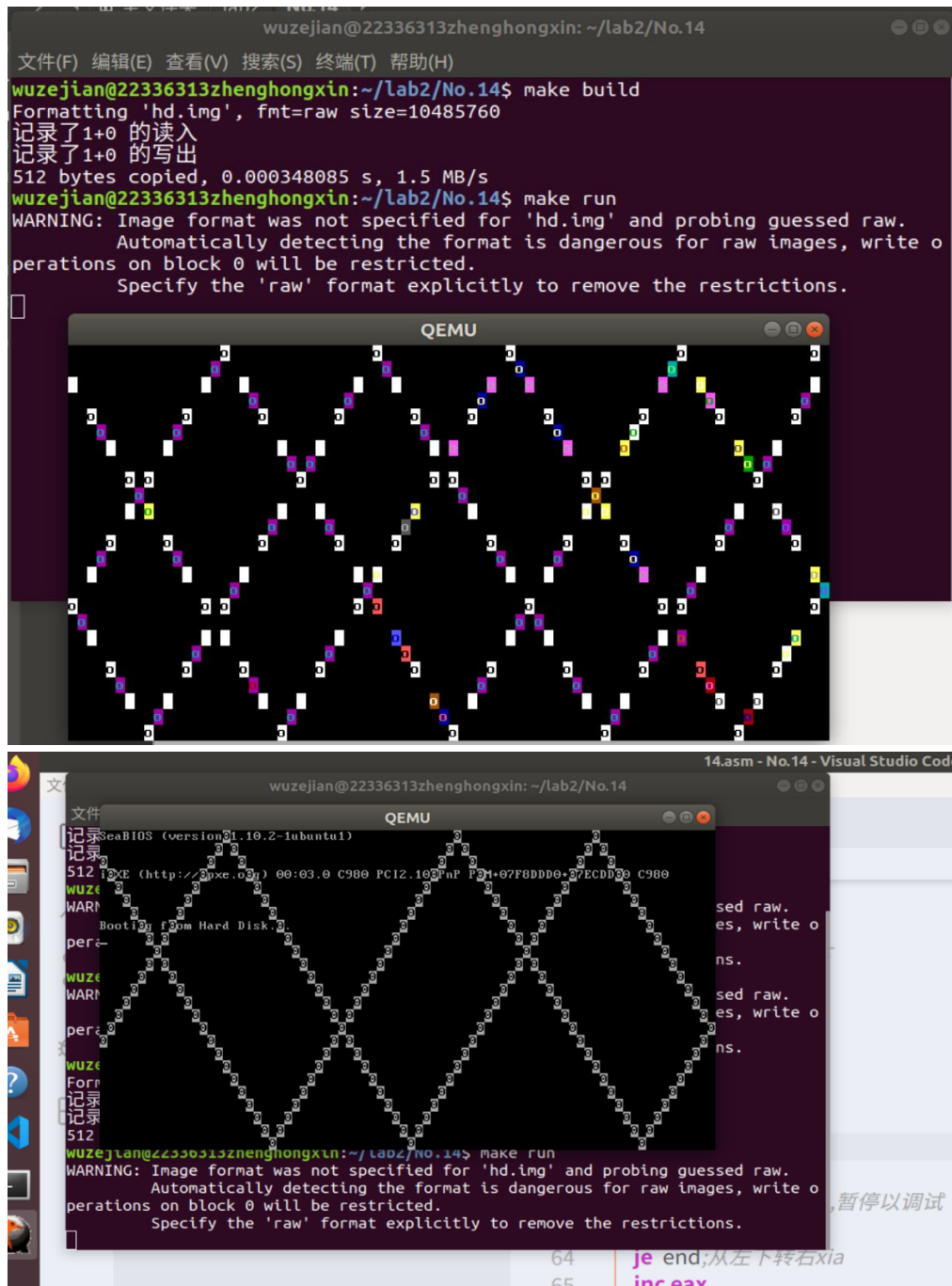
● times 510-(\$-\$\$) db 0

dw 0xaa55

- 实验结果展示：通过执行前述代码，可得下图结果。

同样利用 makefile 文件方便进行操作，通过编译运行后结果如下：

需要注意的是：界面中字符弹射显得似乎不连续而是间断的，实际上并不是，而是在更改背景色和前景色的过程中总会偶尔出现前景色与背景色都为黑色的情况，与界面融为一体了。（另附一张不变色的程序截图）



Section 5 实验总结与心得体会

遇到的问题:

1. 不熟悉汇编语言的语法，经常习惯性的将高级语言的语法用在程序中导致汇编程序报错。
2. 对于重要寄存器的值应该记得入栈和出栈保存，否则会在操作后已经被修改导致不是期望的值，这点在 14 题中尤为明显，因为我使用 `eax` 来存储横坐标，用 `ebx` 来存储纵坐标，用 `ecx` 来存储弹射状态，但是程序调试时经常出现

坐标乱跑甚至全屏打印之类的错误，经检查发现需要用出入栈保证这些重要的数值不会被其他操作修改。

3. 对于第 14 题还遇到一个问题是程序需要每隔一段时间显示下一个字符，而程序运行时间过快，总是一次性打印全部字符，一开始我尝试每显示一个字符后用一个 32 位 1 的二进制数进行自减运算直到 0 来延迟代码的运行，但是由于计算机运行速度过快，效果不明显。后来在与同学交流后了解到可以使用中断机制实现每次延迟 0.5s 的效果。

总结：

本次实验学习了 x86 汇编语言设计，计算机的启动过程，还有了解了 qemu 的字符显示原理。通过课后练习题，熟练了汇编语言程序设计与栈的用法，也提高了对代码的 debug 能力。通过实模式中断的学习，了解了如何使用中断实现特定功能。

Section 6 对实验的改进建议和意见

本实验编写程序过程中，在大多数情况下，编写的程序不一定会一次就成功，需要反复多次地 debug。如果我们每做一次调整，就要输入上面一大堆命令，这无疑会大大降低我们的开发效率。于是我参考了如下资料：

[appendix/debug_with_gdb_and_qemu/README.md · SYSU_2024_OSTA/SYSU-2024-](#)

[Spring-Operating-System - Gitee.com](#)，自行编写了一个 makefile 文件，使得适用

16 题的 3 个小题和 14 题，每次只需要修改文件名即可通用，这样在每次调试代码的时候，只需要使用 make build 和 make run 即可进行编译运行。

Make file 代码如下：（以 16.1.asm 为例）

```
run:
    @qemu-system-i386 -hda hd.img -serial null -parallel stdio
debug:
    @nasm -o 16.1.o -g -f elf32 16.1.asm
    @ld -o 16.1.symbol -melf_i386 -N 16.1.o -Ttext 0x7c00
    @qemu-system-i386 -hda hd.img -s -S -parallel stdio -serial
null
build:
    @nasm -f bin 16.1.asm -o 16.1.bin
    @qemu-img create hd.img 10m
    @dd if=16.1.bin of=hd.img bs=512 count=1 seek=0 conv=notrunc
clean:
    @rm -fr *.bin *.o
```

Section 7 附录：参考资料清单

1. 实验参考书网址：[SYSU-2023-Spring-Operating-System: 中山大学 2023 学年春季操作系统课程 - Gitee.com](#)
2. 参考书附录关于 makefile 的使用：
[appendix/debug_with_gdb_and_qemu/README.md · SYSU 2024_OSTA/SYSU-2024-Spring-Operating-System - Gitee.com](#)
3. 键盘 I/O 中断参考博客：[键盘 I/O 中断调用 \(INT 16H\) 和常见的 int 17H、int 1A H 微机原理 int 16h 的中断号-CSDN 博客](#)
4. 汇编语言编写：[X86 汇编快速入门 - jiftle - 博客园 \(cnblogs.com\)](#)