



# 本科生实验报告

实验课程：\_\_\_\_ 操作系统 \_\_\_\_\_

实验名称：\_\_\_\_ Lab2 \_\_\_\_\_

专业名称：\_\_\_\_ 信息与计算机科学 \_\_\_\_\_

学生姓名：\_\_\_\_ 张文沁 \_\_\_\_\_

学生学号：\_\_\_\_ 20337268 \_\_\_\_\_

实验地点：\_\_\_\_ 无固定地点 \_\_\_\_\_

实验成绩：\_\_\_\_\_

报告时间：\_\_\_\_ 2022. 3. 6 \_\_\_\_\_

## 1. 实验要求

熟悉 32 位 Intel 汇编语言的基本语法；

熟悉汇编语言的编译、链接过程；

掌握简单编程；

掌握磁盘、显示 I/O 操作；

掌握汇编程序调用；

掌握 C 语言与汇编语言之间的调用；

1. 利用 32 位汇编程序从一个数组中找出最大的数（算法不限）；
2. 利用 32 位汇编计算前 20 个 Fibonacci 数列；
3. 利用汇编清理屏幕的显示内容并定位光标；
4. 利用 C 语言和汇编语言实现磁盘文件的拷贝以及显示体现 C 语言与汇编语言之间的调用关系
5. 利用不同颜色显式字符串（可选）；

## 2. 实验过程

a) 利用 32 位汇编程序从一个数组中找出最大的数

- i. 学习 nasm 汇编语言基础语法
- ii. 编写 C 程序
- iii. 翻译成汇编语言
- iv. 调试

b) 利用 32 位汇编计算前 20 个 Fibonacci 数列

- i. 同上

c) 利用汇编清理屏幕的显示内容并定位光标

- i. 参考博客完成
- ii. 需要的工具：qemu
- iii. 原理：通过 BIOS 中断，来滚动屏幕，达到清屏的效果
- iv. 光标位置：通过中断 int 10h 可以在文本坐标下设置光标位置

d) 利用 C 语言和汇编语言实现磁盘文件的拷贝以及显示体现 C 语言与汇编语言之间的调用关系

言与汇编语言之间的调用关系

- i. 参考博客完成，但是只完成了磁盘文件的拷贝和显示
- ii. 代码非本人完成，只是尝试了解了代码逻辑和实现方式
- iii. 读取硬盘
- iv. 逻辑：
  - 1) sector count 寄存器寄存器写入读取的扇区数
  - 2) LBA low 寄存器，LBA mid 寄存器，LBA high 寄存器写入 lba 地址
  - 3) device 寄存器写入 lba 地址和读取模式

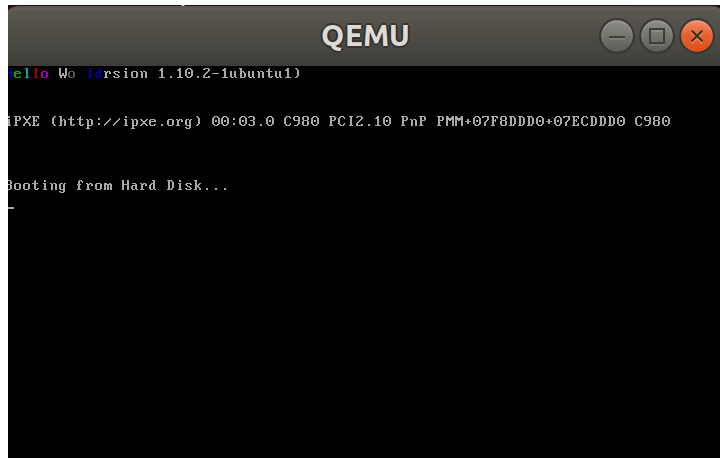
- 4) command 寄存器写入写入命令
- 5) 读取两个字节数据，多次循环直到读取完扇区数据。

e) 利用不同颜色显式字符串：

- i. 了解原理
- ii. 编写代码
- iii. 编译
- iv. 调动 qemu

```
adria@adria-VirtualBox:~/lab2$ touch display.asm
adria@adria-VirtualBox:~/lab2$ nasm -f bin display.asm -o display.bin
adria@adria-VirtualBox:~/lab2$ qemu-img create rgb.img 10m
Formatting 'rgb.img', fmt=raw size=10485760
```

```
dd: 打开 'rgb.bin' 失败: 没有那个文件或目录
adria@adria-VirtualBox:~/lab2$ dd if=display.bin of=rgb.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, 0.000127694 s, 4.0 MB/s
adria@adria-VirtualBox:~/lab2$ qemu-system-i386 -hda rgb.img -serial null -
parallel stdio
WARNING: Image format was not specified for 'rgb.img' and probing guessed r
aw.
Automatically detecting the format is dangerous for raw images, wr
ite operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
```



### 3. 关键代码

a) 利用 32 位汇编程序从一个数组中找出最大的数

1. extern printf
2. extern exit
3. section .data
4. array dd 1,2,3,4,5,6,7,8,9,10
5. format db "%d"
6. section .text

```

7.    global _start
8.    _start:
9.        mov eax,0 ; 记录最大值
10.       mov ebx,array ; 将第一个数字传入 ebx
11.       mov ecx,10 ; ecx 记录数组长度，即循环次数
12.       Loop:
13.           cmp [ebx],eax
14.           jl next ; 大于就跳转
15.           mov eax,[ebx] ; 否则更新数字
16.       next:
17.           add ebx,4 ; 到下一个数字，偏移寻址
18.           loop Loop ; 继续循环
19.       push eax
20.       push format
21.       call printf
22.       push 0
23.       call exit

```

b) 利用 32 位汇编计算前 20 个 Fibonacci 数列

```

1.    %include 'functions.s'
2.
3.    section.text
4.    global _start
5.    _start:
6.
7.        mov ecx, 20
8.        mov eax, 1
9.        mov ebx, 1
10.
11.    fibonacci:
12.        call iprintLF
13.        add ebx, eax
14.        mov edx, eax
15.        mov eax, ebx
16.        mov ebx, edx
17.        dec ecx
18.        jnz fibonacci
19.        call quit
20.    section.data

```

c) 利用汇编清理屏幕的显示内容并定位光标

i. 光标初始化

```
1. Init_Cursor: ; 光标位置初始化
2.     mov ah,0x02
3.     mov bh,0
4.     mov dx,0
5.     int 0x10
```

ii. 清屏

```
1. Clear_Screen: ;清除屏幕
2.     mov ah,0x06
3.     mov al,0
4.     mov cx,0
5.     mov df,0xffff
6.     mov bh,0x17 ;属性为蓝底白字
7.     int 0x10
```

d) 利用 C 语言和汇编语言实现磁盘文件的拷贝以及显示体现 C 语言与汇编语言之间的调用关系

读取磁盘

```
1. .check_status:;检查磁盘状态
2.     nop
3.     in al,dx
4.     and al,0x88 ;第 4 位为 1 表示硬盘准备好数据传输,第
7 位为 1 表示硬盘忙
5.     cmp al,0x08
6.     jnz .check_status ;磁盘数据没准备好,继续循环检查
7.
8.
9.
10.    ;设置循环次数到 cx
11.    mov ax,cx ;乘法 ax 存放目标操作数
12.    mov dx,256
13.    mul dx
14.    mov cx,ax ;循环次数 = 扇区数 x 512 / 2
15.    mov bx,di
16.    mov dx,0x1F0
17.
18. .read_data:
19.     in ax,dx ;读取数据
20.     mov [bx],ax ;复制数据到内存
21.     add bx,2 ;读取完成,内存地址后移 2 个字节
22.
23.     loop .read_data
24.     ret
```

## 输出字符串：

```
1.      ;程序核心内容
2.      Entry:
3.      ;-----
4.      ;输出字符串
5.      mov si,HelloMsg      ;将 HelloMsg 的地址放入 si
6.      mov dh,0             ;设置显示行
7.      mov dl,0             ;设置显示列
8.      call Func_Sprint     ;调用函数
9.      jmp $                ;让 CPU 挂起，等待指令
```

e) 利用不同颜色显式字符串：

编译指令：

```
1.      nasm -f bin mbr.asm -o mbr.bin //-f 输出的文件格式，-o 输出的文件名。
      mbr.bin 保存机器指令
2.      qemu-img create filename [size] //filename 生成的硬盘文件名
3.      qemu-img create hd.img 10m
4.      dd if=mbr.bin of=hd.img bs=512 count=1 seek=0 conv=notrunc
5.      //if 输入文件。
6.      //of 输出文件。
7.      //bs 表示块大小，以字节表示。
8.      //count 表示写入的块数目。
9.      //seek 表示越过输出文件中多少块之后再写入。
10.     //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。
11.     qemu-system-i386 -hda hd.img -serial null -parallel stdio
12.     //-hda hd.img 表示将文件 hd.img 作为第 0 号磁盘映像。
13.     //-serial dev 表示重定向虚拟串口到空设备中。
14.     //-parallel stdio 表示重定向虚拟并口到主机标准输入输出设备中。
```

汇编代码：

```
1.      org 0x7c00 ;代码中的代码标号和数据标号从 0x7c00 开始
2.      [bits 16] ;代码为 16 位格式
3.      xor ax, ax ; ax 初始化为 0
4.      ; 初始化段寄存器，段地址全部设为 0
5.      mov ds, ax
6.      ; 初始化栈指针
7.      mov sp, 0x7c00 ;
8.      mov ax, 0xb800
9.      mov gs, ax
10.     mov ah, 0x01 ;蓝色
```

```

11.  mov al, 'H'
12.  mov [gs:2 * 0], ax
13.  mov ah, 0x02 ;绿色
14.  mov al, 'e'
15.  mov [gs:2 * 1], ax
16.  mov ah, 0x03 ;浅蓝色
17.  mov al, 'l'
18.  mov [gs:2 * 2], ax
19.  mov ah, 0x04 ;红色
20.  mov al, 'l'
21.  mov [gs:2 * 3], ax
22.  mov ah, 0x05 ;粉色
23.  mov al, 'o'
24.  mov [gs:2 * 4], ax
25.  mov ah, 0x06
26.  mov al, ' '
27.  mov [gs:2 * 5], ax
28.  mov ah, 0x07 ;白色
29.  mov al, 'W'
30.  mov [gs:2 * 6], ax
31.  mov ah, 0x08 ;灰色
32.  mov al, 'o'
33.  mov [gs:2 * 7], ax
34.  mov ah, 0x00
35.  mov al, 'r' ;黑色
36.  mov [gs:2 * 8], ax
37.  mov ah, 0x01
38.  mov al, 'l'
39.  mov [gs:2 * 9], ax
40.  mov al, 'd'
41.  mov [gs:2 * 10], ax
42.  jmp $ ; 死循环
43.  times 510 - ($ - $$) db 0
44.  db 0x55, 0xaa

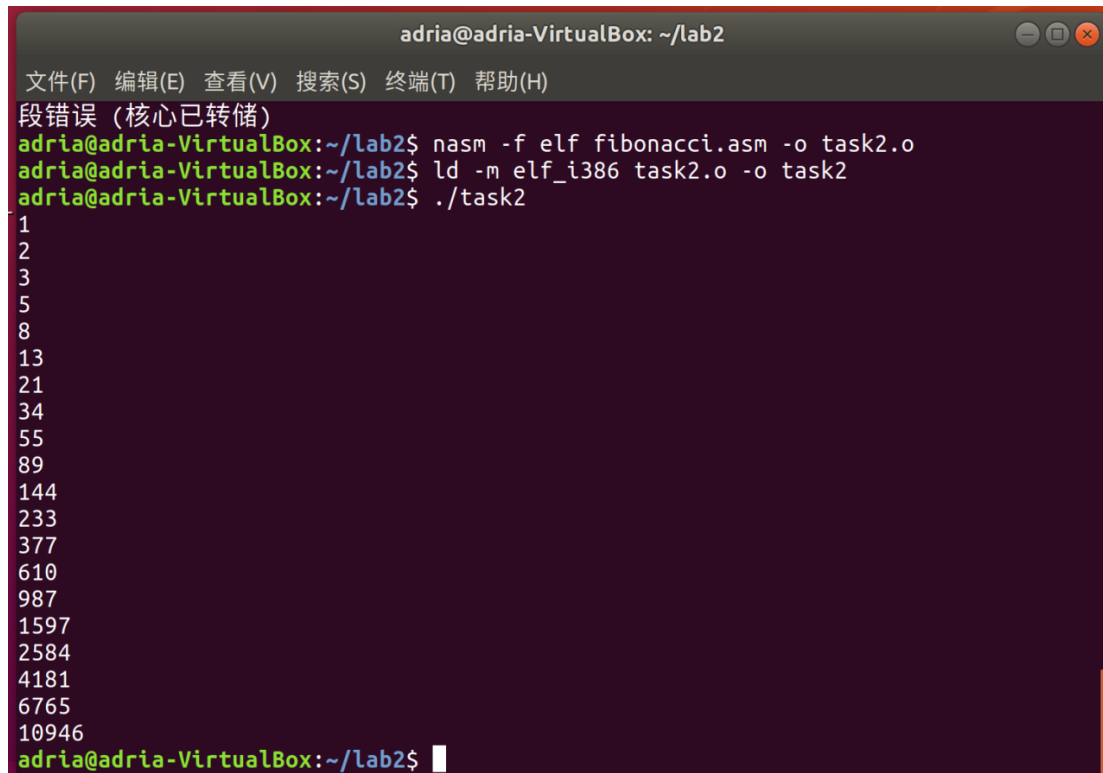
```

## 4. 实验结果

a) 利用 32 位汇编程序从一个数组中找出最大的数  
 输入的数组为 1-10，输出为 10

```
adria@adria-VirtualBox:~/lab2$ nasm -f elf32 largest.s
adria@adria-VirtualBox:~/lab2$ ld -m elf_i386 largest.o -o largest -lc -I /lib/ld-linux.so.2
adria@adria-VirtualBox:~/lab2$ ./largest
10adria@adria-VirtualBox:~/lab2$
```

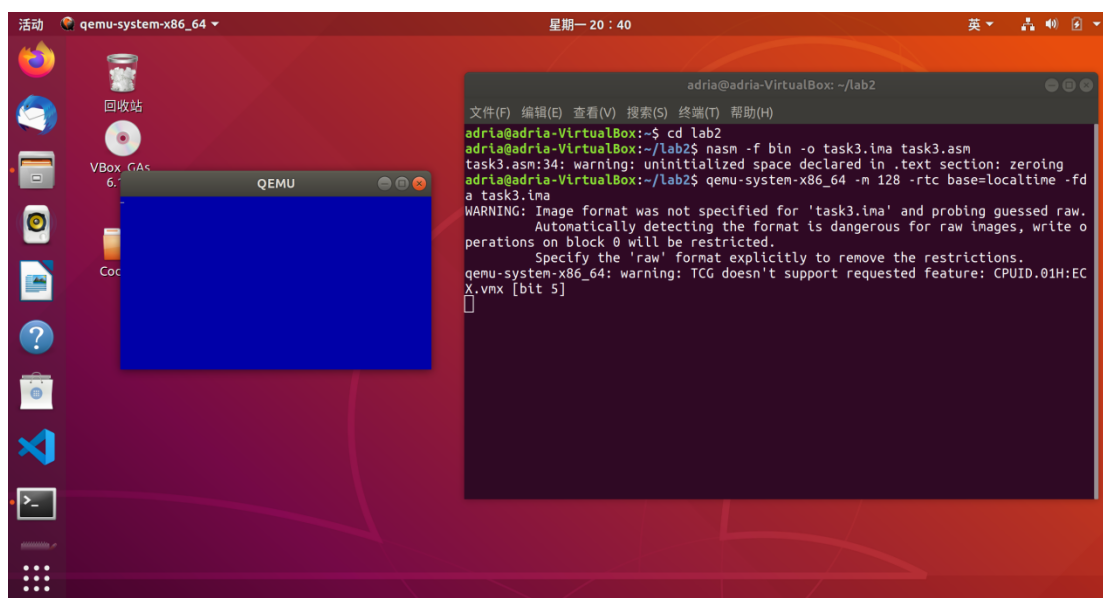
b) 利用 32 位汇编计算前 20 个 Fibonacci 数列



```
adria@adria-VirtualBox: ~/lab2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
段错误 (核心已转储)
adria@adria-VirtualBox:~/lab2$ nasm -f elf fibonacci.asm -o task2.o
adria@adria-VirtualBox:~/lab2$ ld -m elf_i386 task2.o -o task2
adria@adria-VirtualBox:~/lab2$ ./task2
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
adria@adria-VirtualBox:~/lab2$
```

c) 利用汇编清理屏幕的显示内容并定位光标

如图所示，用 qemu 完成了清屏



```
adria@adria-VirtualBox: ~/lab2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
adria@adria-VirtualBox:~/lab2$ cd lab2
adria@adria-VirtualBox:~/lab2$ nasm -f bin -o task3.ima task3.asm
task3.asm:34: warning: uninitialized space declared in .text section: zeroing
adria@adria-VirtualBox:~/lab2$ qemu-system-x86_64 -m 128 -rtc base=localtime -fd a task3.ima
WARNING: Image format was not specified for 'task3.ima' 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-system-x86_64: warning: TCG doesn't support requested feature: CPUID.01H:ECX.vmx [bit 5]
```

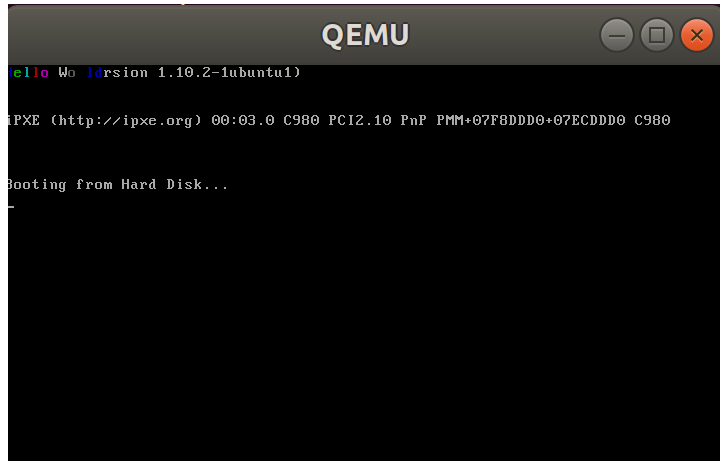
d) 利用 C 语言和汇编语言实现磁盘文件的拷贝以及显示体现 C 语



## 言与汇编语言之间的调用关系

输出蓝底白字 “Hello world”

e) 利用不同颜色显式字符串：



## 5. 总结

- a) 利用 32 位汇编程序从一个数组中找出最大的数
    - i. 汇编语言分很多种，nasm 和其他版本的语法相去甚远，查找资料的过程中要注意分辨
    - ii. 连接 C 库的时候所用的指令需要加上 `-I /lib/ld-linux.so.2`
    - iii. `Mov ecx, 10` 这段代码一定需要出现
  - b) 利用 32 位汇编计算前 20 个 Fibonacci 数列
    - i. 逻辑很简单，但是调用 C 函数总是无法开始循环，只输出三个数字便会结束循环。
    - ii. 查询之后发现可能是 `ecx` 和 `printf` 有牵连关系，导致无法完成循环，故使用了教程网站中的头文件中的输出函数
  - c) 跟着博客和实验指导完成了任务，大致了解了一些中断类型及其用处，题目中用 `int 10h` 完成了中断。考虑并发性、用户态到核心态的转换依赖于中断完成。
6. 参考资料：
- a) <https://asmtutor.com/>
  - b) <https://www.cnblogs.com/mlzrq/p/10223028.html>
  - c) <https://www.cnblogs.com/mlzrq/p/10223060.html>