

# 南京航空航天大学 计算机科学与技术学院

## 汇编语言 课程实验

### 目录

南京航空航天大学 计算机科学与技术学院 汇编语言 课程实验 .....	1
一、 I/O 与中断学习笔记 .....	2
1.1 基本概念 .....	2
1.2 I/O 编址方式 .....	2
1.3 I/O 控制方式 .....	4
1.4 中断的过程 .....	4
1.5 中断服务程序设计 .....	4
二、实验介绍 .....	6
三、实验展示 .....	7
3.1 第七章程序 .....	8
3.2 第八章程序 .....	14
3.3 Bochs 环境下实现 Pacman .....	17
3.4 VirtualBox 环境下集成的 Pacman .....	18
四、实现思路 .....	19
4.1 整体架构 .....	19
4.2 中断服务程序 .....	20
4.3 吃豆豆小游戏 .....	22
附录：考试压轴题 .....	24
参考文献 .....	27

文档说明(README.md):

- 本实验基于第七章的加载器 `myloader` 完成。
- 附录是考试的最后一道大题（输入有符号十进制数，排序并输出），考试时间紧张，未能完整完成该题目，课后基于 `Emu8086` 的环境完成。
- 文档阅读顺序：实验介绍 -> 实验展示 -> 心得体会 -> 教学建议。

# 一、I/O 与中断学习笔记

## 1.1 基本概念

1. I/O 端口与存储器接口：CPU 与外部设备、存储器的连接和数据交换都需要通过接口设备来实现，前者被称为 I/O 端口，而后者则被称为存储器接口。
2. 硬件接口：实质是逻辑控制电路，完成输入/输出的转换。从程序员的层面来看，就是一组寄存器和内存单元。
3. IN/OUT 指令：略。
4. 中断与异常的区别：外部设备“打断”CPU 是中断；CPU 自身内部出错称为异常（如 DIV 指令溢出错误）；中断用于处理异步发生的外部事件；异常用于处理同步发生的内部事件。
5. 中断向量：即中断服务程序的入口地址，每个中断向量 4 个字节，高 2 字节是中断服务程序的段值，低 2 字节是段内偏移。每个中断向量都有唯一的中断类型号。
6. 中断向量表：最多含有 256 个中断服务程序，放在内存地址 00000H - 003FFH 的内存空间（1KB 大小）。

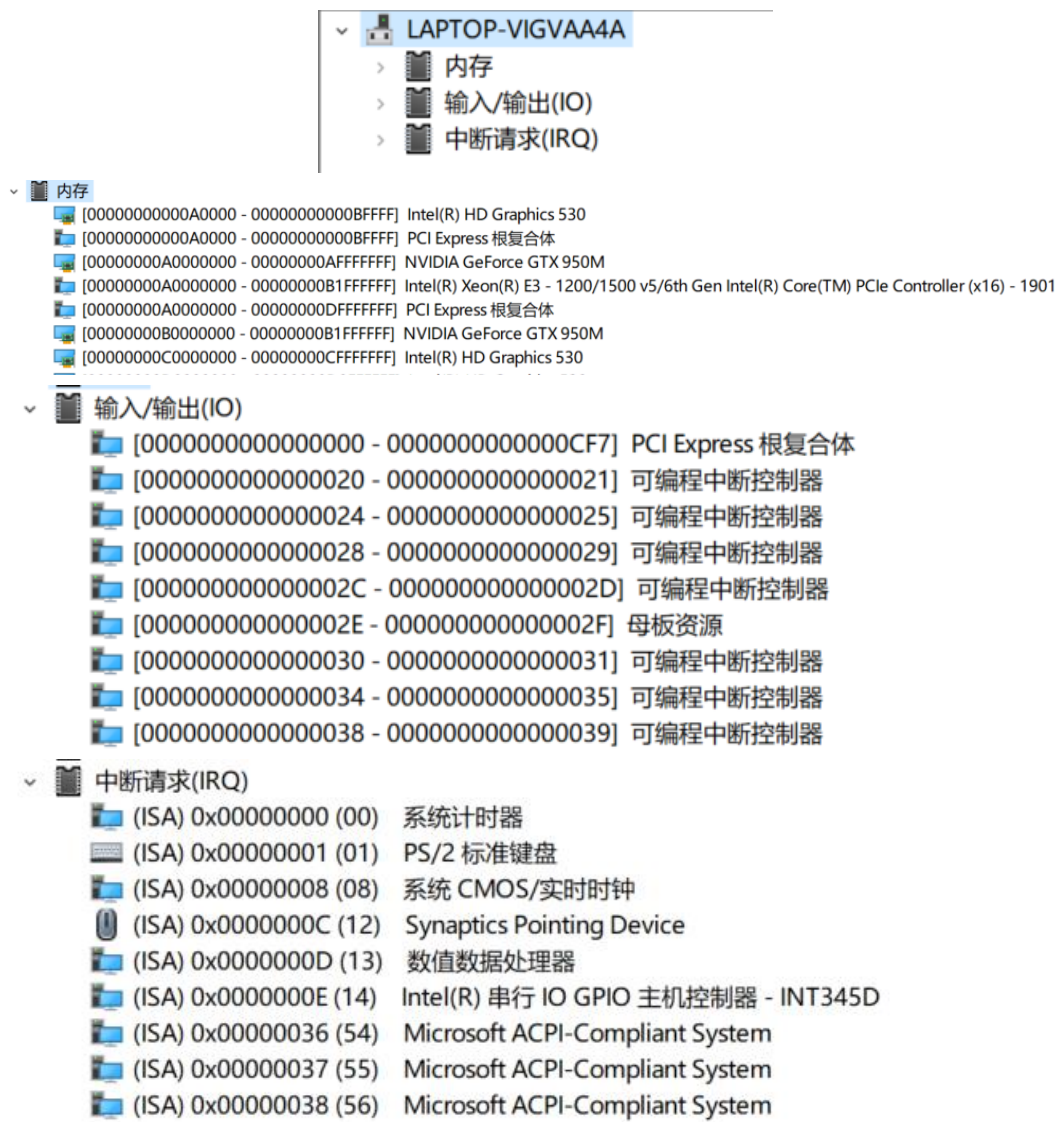
## 1.2 I/O 编址方式

两种基本方法：存储器映射编址（统一编址）和 I/O 映射编址（独立编址）。两种方法的对比如下：

	存储器映射编址	I/O 映射编址
含义	把 I/O 看作是一个内存单元，统一安排在内存单元中	I/O 设备有一套独立的地址空间

该方式下访问 I/O 的指令	访存指令访问	IN/OUT 指令
优点	对外部设备可使用全部的访存指令，使用方便	1. 不占用内存单元，节省内存空间 2. 指令执行速度快（无须访存） 3. 编程时易于区分 I/O 与内存
缺点	1. 占用了内存空间 2. 速度慢（访存次数多）	需要设计复杂的控制电路

独立编址是目前 PC 广泛使用的 I/O 编址方式。在自己的电脑上也能够查找每个 I/O 端口对应的外部设备信息。如下图：



其中内存地址范围是：0H – FFFFFFFFH（4GB 内存）；I/O 的地址 0 – FFFFH（64K 地址空间），其中中断请求 IRQ 也拥有一套独立的编址空间，这一点留在后面说明。

## 1.3 I/O 控制方式

对 I/O 外部设备的控制方式主要有三种：直接程序控制（包括无条件传送方式和查询状态方式）、中断控制、DMA 控制。本次实验的重点内容就是“中断”，因此只对中断控制方式进行叙述，其他可参考《微机原理与接口技术》第六章。

### 1. 中断控制方式

在中断方式下，CPU 与 I/O 设备均在独立工作，在外设状态满足要求时，外设通过 I/O 端口向 CPU 发送请求信号；接到请求信号后，CPU 才去响应，执行 I/O 设备的输入或输出操作。

## 1.4 中断的过程

中断过程分为五个阶段：中断请求，中断判优，中断响应，中断处理，中断返回。

1. 请求：外部设备通过可编程中断控制器（集成芯片 8259）向 IRQ 引脚发送中断的请求。
2. 判断：若有多个中断请求，则需要对他们判断优先级，然后再进行处理。（判优的准则有多种方式，如全嵌套方式、自动循环方式，可查阅《微机原理与接口技术》）
3. 响应：处理器向外设发送中断已被响应的信号。包括以下几个阶段：接收中断请求、中断响应周期、CS+IP/EIP/RIP 压栈、取中断类型号 N。
4. 处理：该过程包括：保护现场（PUSH 指令）、开中断（STI 指令）、执行中断服务程序（CALL 指令）、关中断（CLI 指令）、恢复现场（POPA 指令）。
5. 返回：返回处理器执行中断前的下一条指令的位置（RETF 指令）。

## 1.5 中断服务程序设计

要想自己设计一个中断服务程序并运行，总结出以下几个步骤：

1. 先实现自己的中断服务程序，以子程序的形式，记为 `new_intXXH_handler`。代码大致框架如下：

```

;-----
new_intXXH_handler:
    PUSHA                    ;保护现场
    ;
    STI                      ;开中断
    ;
    ;add your code here
    CALL IntXXH_function    ;此处实现中断服务功能
    ;
    CLI                      ;关中断
    ;
    ;中断结束命令
    MOV AL, 20H              ;操纵中断控制器8259
    OUT 20H, AL
    ;
    POPA                     ;恢复现场
    ;
    IRET                     ;中断返回
;-----

```

2. 保存旧 INT XXH 的段值与偏移。oldxxh 是一个 4 字节数据 (DD)，通过以下指令实现：

```

;
MOV SI, XXH*4
MOV AX, 0
MOV ES, AX    ;ES=0

MOV AX, [ES:SI]    ;原中断偏移
MOV [oldxxh], AX
MOV AX, [ES:SI+2]  ;原中断段值
MOV [oldxxh+2], AX

```

3. 设置 new\_INTXXH\_handler 为新的中断服务程序，如下：

```

MOV AX, 0
MOV DS, AX    ;DS=0, 访问中断向量表的内存区域
CLI
MOV WORD [XXH*4], new_INTXXH_handler ;低2字节是段内偏移
MOV [XXH*4+2], CS ;高2字节是段值，即当前的CS寄存器
STI

```

4. RETF 返回之前恢复原中断程序。（如果想 new\_INTXXH\_handler 继续生效则不需要该步骤）

```

restore_XXH_handler:
    MOV AX,0
    MOV ES,AX
    MOV SI, XXH*4
    MOV EAX, [CS:oldxxh]
    MOV [ES:SI], EAX
    RETF

```

## 二、实验介绍

本实验基于第七章的 `myloader` 程序完成，在此基础上加入了自己实现的中断服务程序，程序的运行展示请看第三章节“实验展示”。输入和输出均以**字符串形式**实现。

此外，本次实验先基于 `Nasm + Bochs` 实现了一个“Pacman 吃豆豆”小游戏。`Bochs` 模拟器可单步+断点调试 `Nasm` 汇编程序，避免了 `Nasm+VirtualBox` 这种人肉调试的弱鸡环境。

后期将其集成到第七章的 `Nasm + VirtualBox` 环境下，以进一步展示程序。（这一步其实耗时巨大，因为 `Bochs` 环境与 `Virtual Box` 环境有很大的差异，）

其中，程序 0-6 在第七章已实现，只是稍作修改，以适应本次实验内容。程序 7-13 是本次实验所完成的程序。全部实现的程序如下表格所示：

序号	程序名称	程序功能描述
0	Myloader	运行在裸机上的加载器，负责管理和加载磁盘上的程序。同时也是一个简单的用户界面，实现 2 个简单指令： <code>list</code> 指令展示程序列表； <code>clsr</code> 指令清空屏幕。
1	PragramLit	打印可运行的程序列表
2	HelloWorld	简单的 HelloWorld 程序，作为实现加载器的入门程序
3	Add Calculator	一个简单加法运算程序：输入 A,B 两个十进制数（不得超过 $2^{32}$ ），输出 A+B 的和。
4	ShowSeg	加载器入门程序：展示程序运行时的段值
5	ASCII2HEX	第七章集成的程序：输出一个 ASCII 可见字符的十六进制字符串

6	ShowMemory	显示裸机上 0xF000 内存的 10 个 8 位数,分别以 10 进制和 16 进制字符串输出,亦可自定义为其他内存地址。
7	ReadCMOS	通过访问 CMOS, 获取系统的实时时间, 并以 10 进制字符串形式打印。操作方式: 按一次空格显示一次时间, 按回车结束。
8	Int00H - divide Calculator	自定义中断程序: 作为一个简单的除法程序, 输入两个十进制数 A,B (不超过 $2^{32}$ )。若无除法溢出, 则输出 A/B 的商和余数; 若除法溢出 (例如 B=0), 则调用自定义的 INT 00H 中断程序, 提示除法溢出。
9	Int09H-keyboard	自定义中断程序: 限定键盘的响应, 只允许 qwertyuiop 十个按键, 并对其建立 0-9 的映射。每按下一次按键, 输出一个数字。
10	Int1CH - system real clock	自定义中断程序: 基于 ReadCMOS 程序实现, 运行即显示实时系统时间, 并能够动态变化, 不需要再按空格。
11	Int90H-IOExtension	自定义中断程序: 模仿课本程序实现, 实现带属性的 TTY 输出方式, 并将该中断程序应用于显示 ProgramList (彩色输出)。
12	Bubble Sort	考试的最后一道大题, 本次实验顺便完成。
13	Pacman Eat Dots Game	一个“吃豆豆”经典小游戏, 先是基于 Nasm + Bochs 实现的; 后期将其集成到 VirtualBox 环境下 (为了更好地展示所有程序)。具体展示请看第三章节。

### 三、实验展示

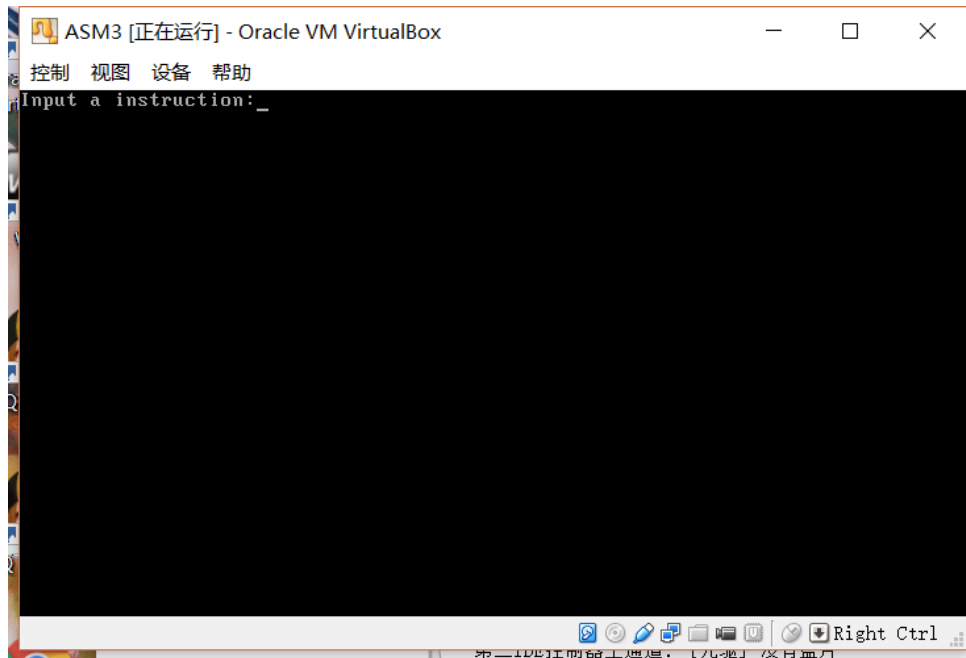
3.1 节“第七章程序”是从第七章实验报告复制过来的, 便于本文档读者理解实验的内容, 也有助于理解本文的 myloader 的总体框架设计。

3.2 节“第八章程序”是在本次实验中新增的程序。

3.3 节和 3.4 节着重介绍一下“吃豆豆”小游戏。（仿照王爽《汇编语言》的贪吃蛇案例，加入个性化总断程序）

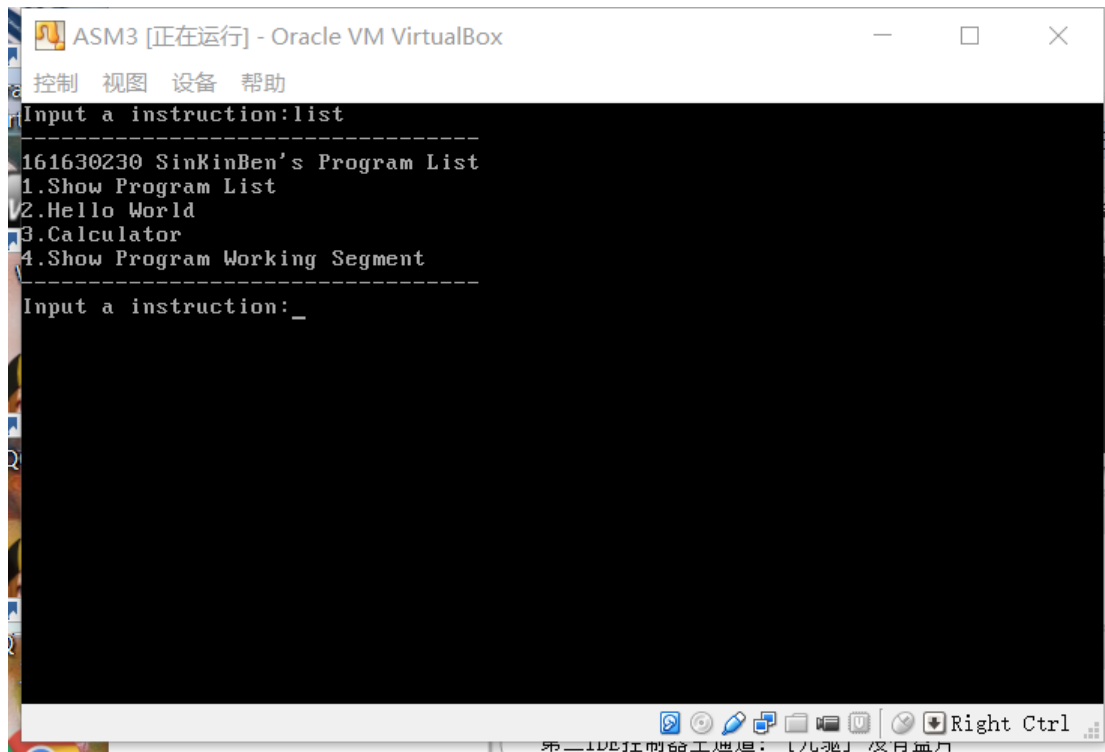
## 3.1 第七章程序

### 1. 初次启动

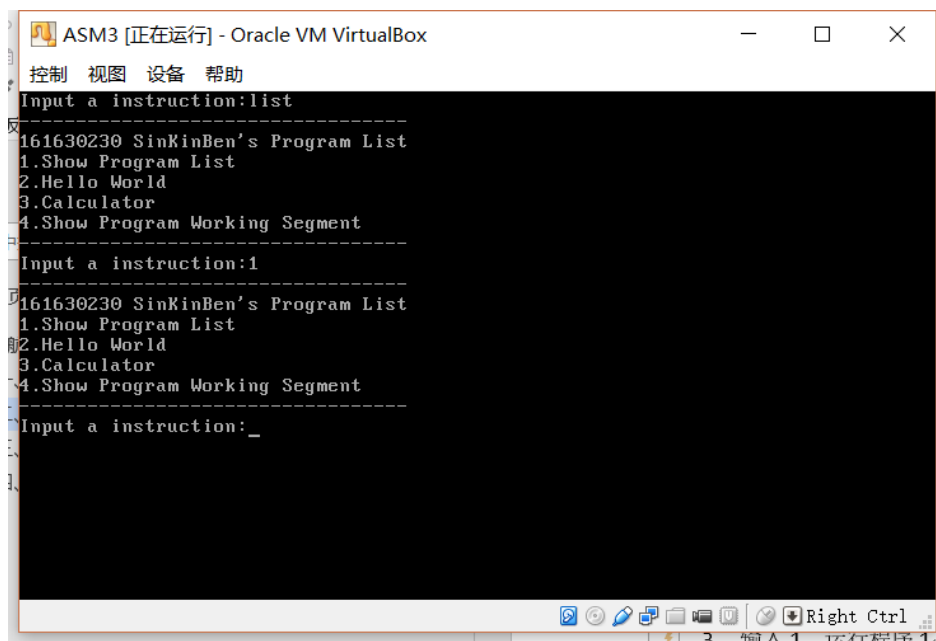


### 2. 输入 list, 显示程序列表

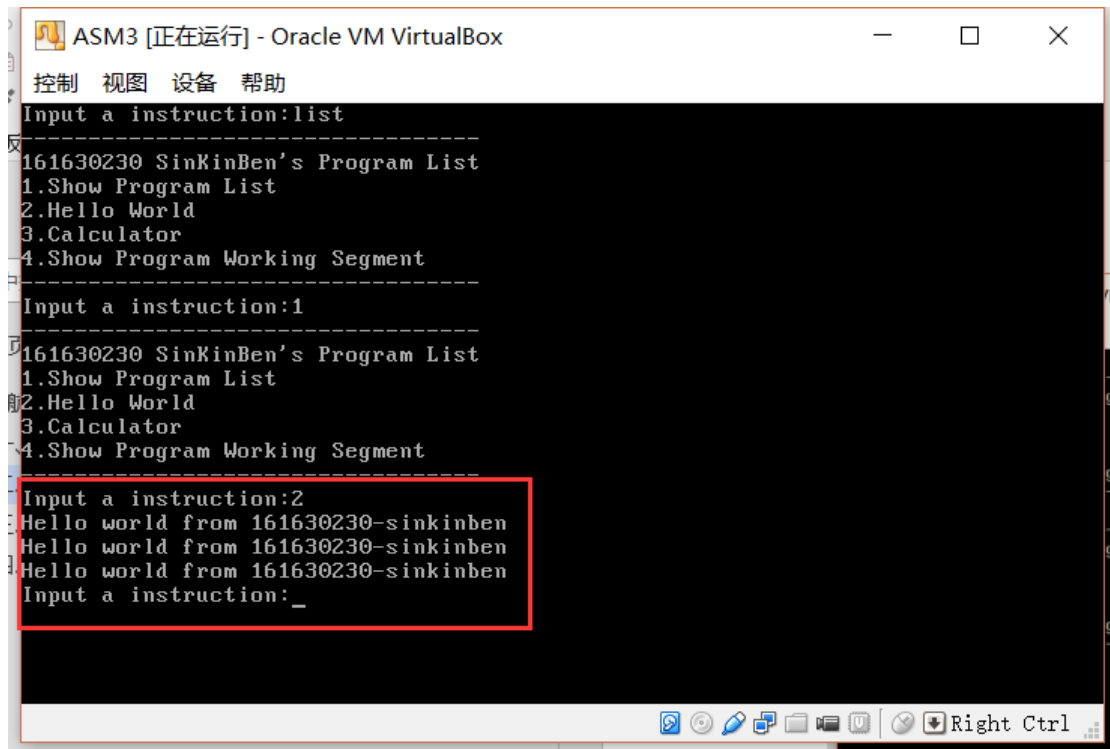




3. 输入 1，运行程序 1



4. 输入 2，运行程序 2 hello world



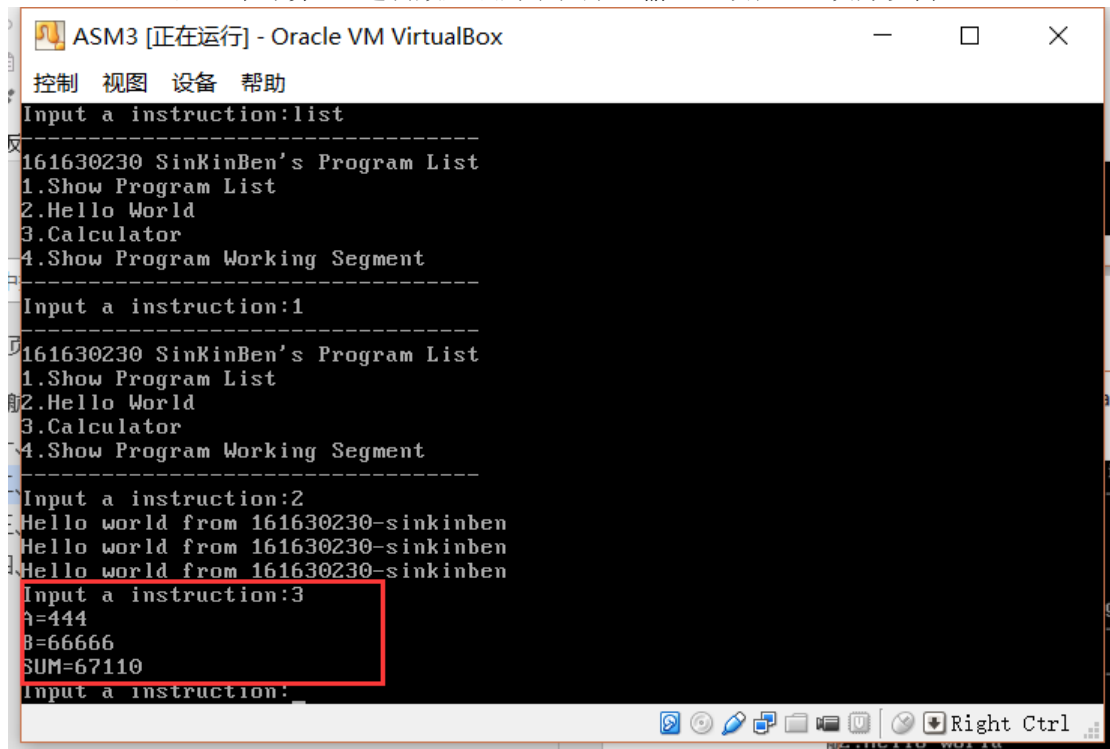
ASM3 [正在运行] - Oracle VM VirtualBox

控制 视图 设备 帮助

```
Input a instruction:list
-----
161630230 SinKinBen's Program List
1.Show Program List
2.Hello World
3.Calculator
4.Show Program Working Segment
-----
Input a instruction:1
-----
161630230 SinKinBen's Program List
1.Show Program List
2.Hello World
3.Calculator
4.Show Program Working Segment
-----
Input a instruction:2
Hello world from 161630230-sinkinben
Hello world from 161630230-sinkinben
Hello world from 161630230-sinkinben
Input a instruction:_
```

5. 输入 3，运行程序 3 calculator

calculator 是一个计算 10 进制数加法的小程序，输入必须在  $2^{32}$  次方以内。



ASM3 [正在运行] - Oracle VM VirtualBox

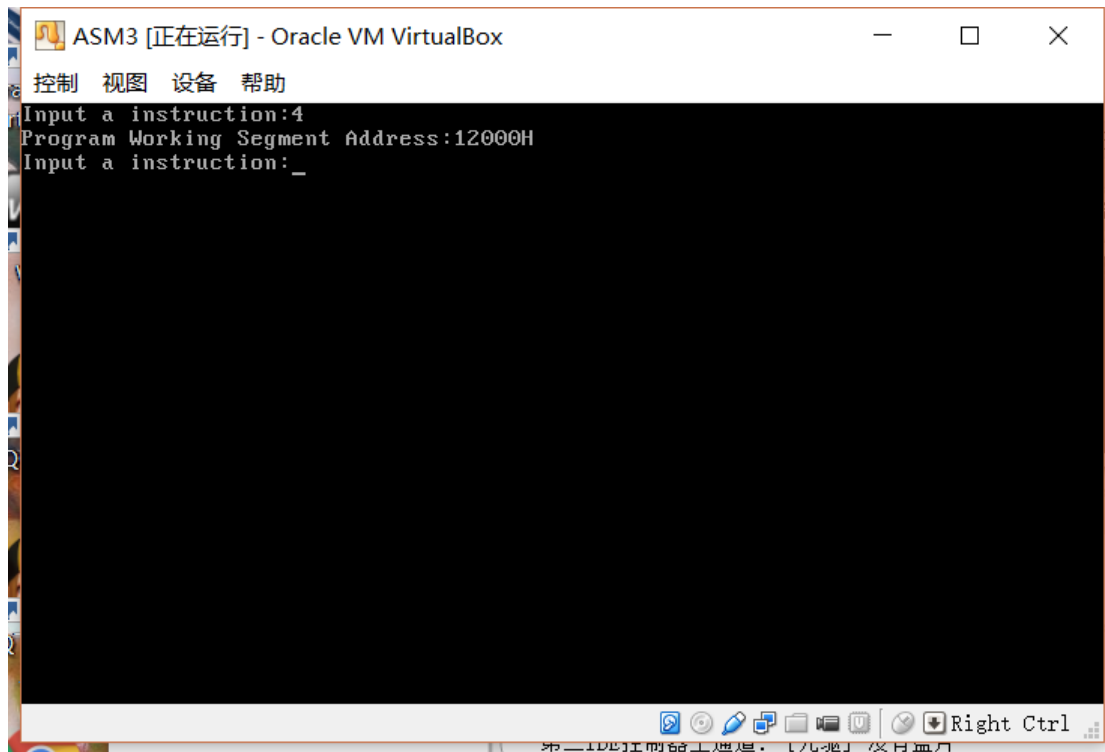
控制 视图 设备 帮助

```
Input a instruction:list
-----
161630230 SinKinBen's Program List
1.Show Program List
2.Hello World
3.Calculator
4.Show Program Working Segment
-----
Input a instruction:1
-----
161630230 SinKinBen's Program List
1.Show Program List
2.Hello World
3.Calculator
4.Show Program Working Segment
-----
Input a instruction:2
Hello world from 161630230-sinkinben
Hello world from 161630230-sinkinben
Hello world from 161630230-sinkinben
Input a instruction:3
a=444
b=66666
SUM=67110
Input a instruction:
-----
```

6. 输入 4，运行程序 4

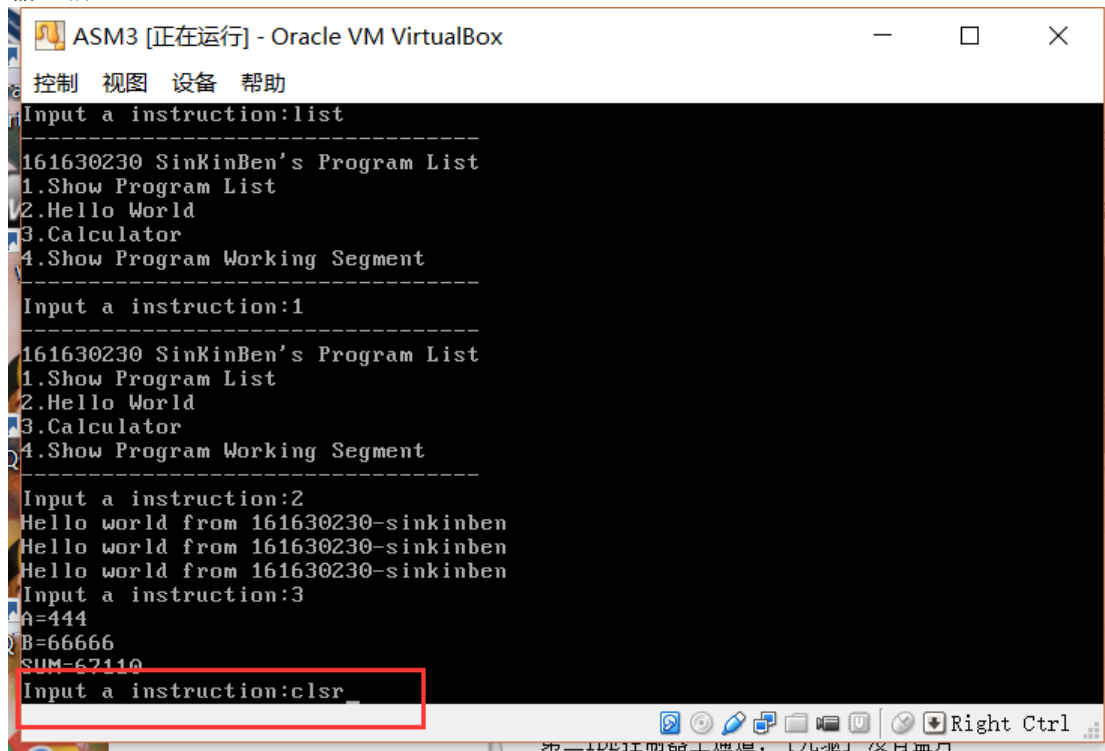
(屏幕信息用了 `clrscr` 清理)

程序 4 是书上的例子，显示这个程序工作的起始段值

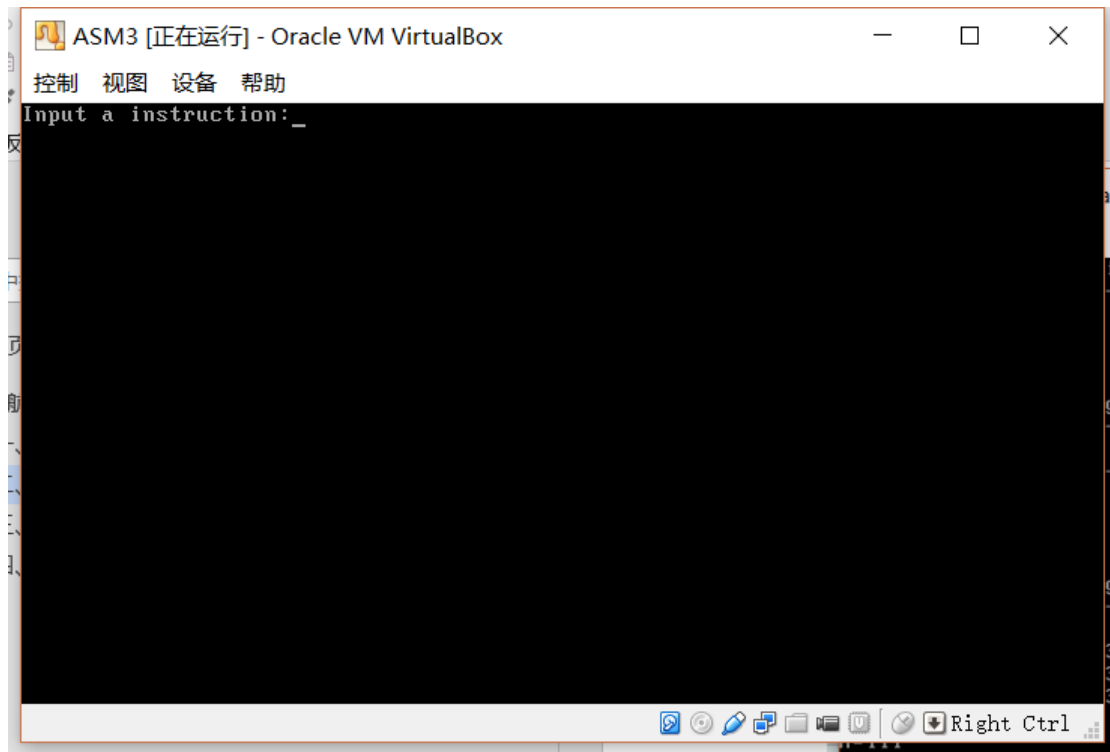


7. 输入 clr 命令，清屏

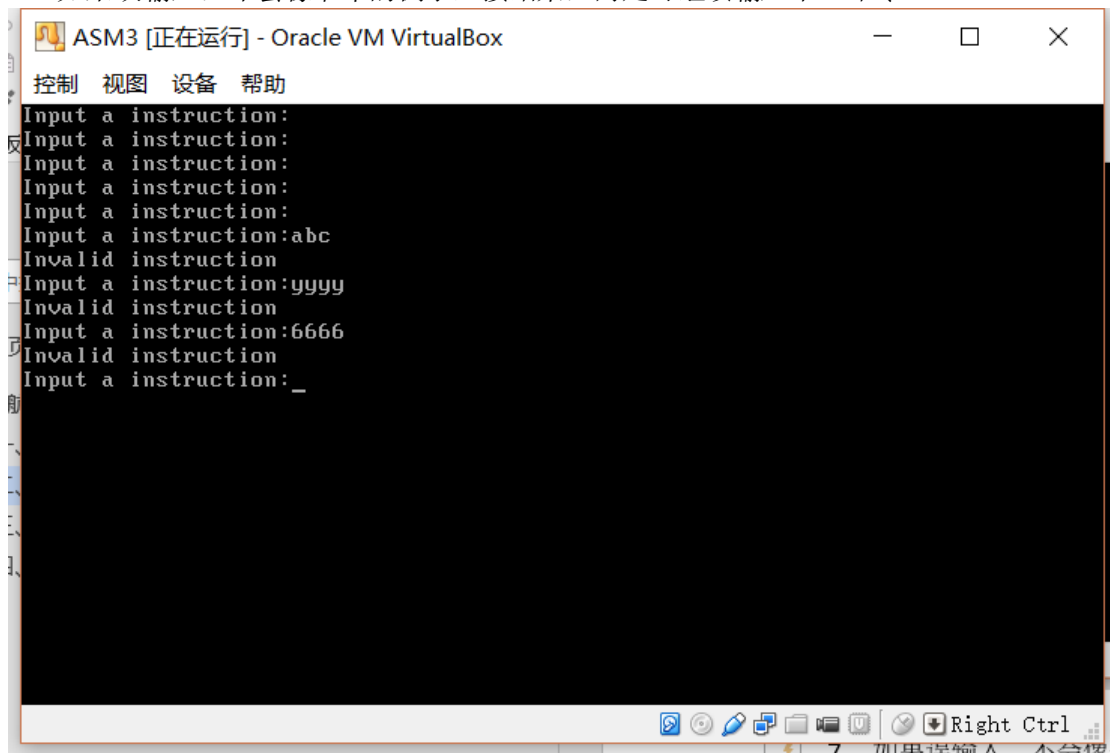
输入前



回车确认



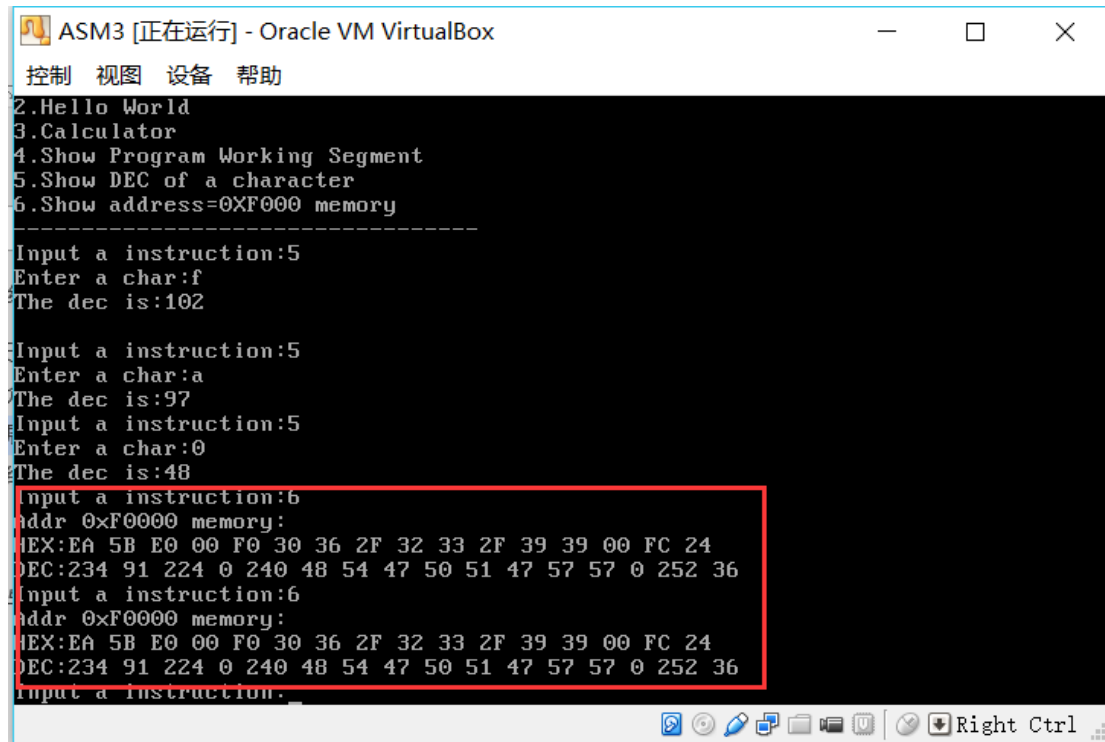
8. 如果误输入，不会像书本的例子直接结束，而是可继续输入下一命令



9. 2018/11/4 更新功能

```
ASM3 [正在运行] - Oracle VM VirtualBox
控制 视图 设备 帮助
Input a instruction:list
-----
161630230 SinKinBen's Program List
1.Show Program List
2.Hello World
3.Calculator
4.Show Program Working Segment
5.Show DEC of a character
6.Show address=0XF000 memory
-----
Input a instruction:_
```

```
ASM3 [正在运行] - Oracle VM VirtualBox
控制 视图 设备 帮助
Input a instruction:list
-----
161630230 SinKinBen's Program List
1.Show Program List
2.Hello World
3.Calculator
4.Show Program Working Segment
5.Show DEC of a character
6.Show address=0XF000 memory
-----
Input a instruction:5
Enter a char:f
The dec is:102
Input a instruction:5
Enter a char:a
The dec is:97
Input a instruction:5
Enter a char:0
The dec is:48
Input a instruction:_
```



```
ASM3 [正在运行] - Oracle VM VirtualBox
控制 视图 设备 帮助
2.Hello World
3.Calculator
4.Show Program Working Segment
5.Show DEC of a character
6.Show address=0xF000 memory
-----
Input a instruction:5
Enter a char:f
The dec is:102

Input a instruction:5
Enter a char:a
The dec is:97

Input a instruction:5
Enter a char:0
The dec is:48

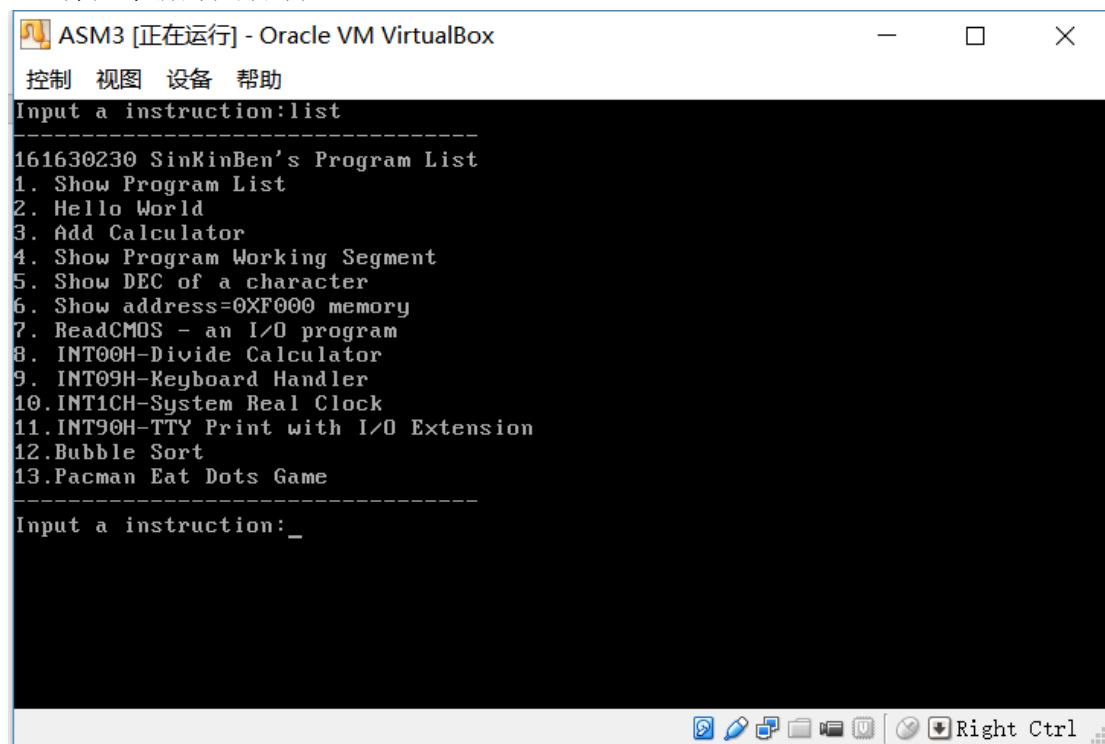
Input a instruction:6
addr 0xF0000 memory:
HEX:EA 5B E0 00 F0 30 36 2F 32 33 2F 39 39 00 FC 24
DEC:234 91 224 0 240 48 54 47 50 51 47 57 57 0 252 36

Input a instruction:6
addr 0xF0000 memory:
HEX:EA 5B E0 00 F0 30 36 2F 32 33 2F 39 39 00 FC 24
DEC:234 91 224 0 240 48 54 47 50 51 47 57 57 0 252 36

Input a instruction:
-----
```

## 3.2 第八章程序

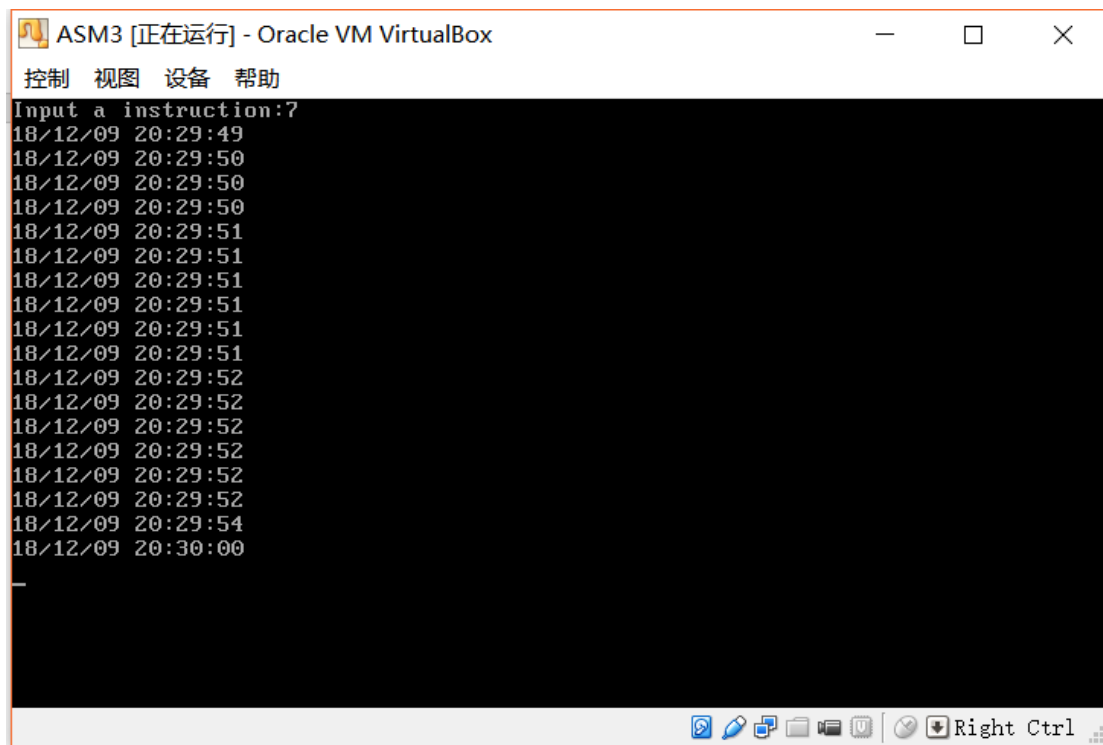
### 1. 第八章新的程序列表



```
ASM3 [正在运行] - Oracle VM VirtualBox
控制 视图 设备 帮助
Input a instruction:list
-----
161630230 SinKinBen's Program List
1. Show Program List
2. Hello World
3. Add Calculator
4. Show Program Working Segment
5. Show DEC of a character
6. Show address=0xF000 memory
7. ReadCMOS - an I/O program
8. INT00H-Divide Calculator
9. INT09H-KeyBoard Handler
10. INT1CH-System Real Clock
11. INT90H-TTY Print with I/O Extension
12. Bubble Sort
13. Pacman Eat Dots Game
-----
Input a instruction:_
```

### 2. ReadCMOS

空格读取多次时间，回车结束



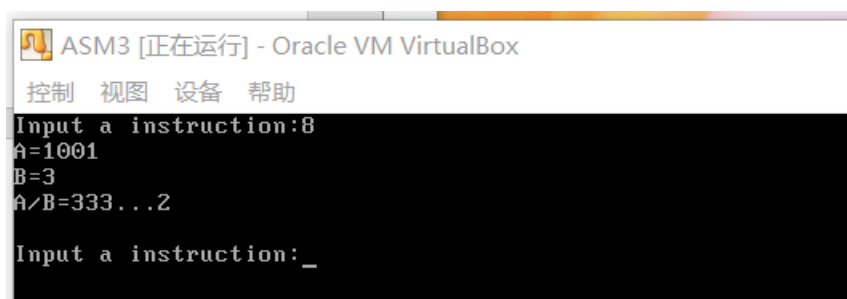
```
ASM3 [正在运行] - Oracle VM VirtualBox
控制 视图 设备 帮助
Input a instruction:?
18/12/09 20:29:49
18/12/09 20:29:50
18/12/09 20:29:50
18/12/09 20:29:50
18/12/09 20:29:51
18/12/09 20:29:51
18/12/09 20:29:51
18/12/09 20:29:51
18/12/09 20:29:51
18/12/09 20:29:51
18/12/09 20:29:52
18/12/09 20:29:52
18/12/09 20:29:52
18/12/09 20:29:52
18/12/09 20:29:52
18/12/09 20:29:52
18/12/09 20:29:54
18/12/09 20:30:00
_
```

### 3. INT00H-Devide Calculator

除法正常，无溢出情况：



```
ASM3 [正在运行] - Oracle VM VirtualBox
控制 视图 设备 帮助
Input a instruction:8
A=8
B=3
A/B=2...2
Input a instruction:_
```



```
ASM3 [正在运行] - Oracle VM VirtualBox
控制 视图 设备 帮助
Input a instruction:8
A=1001
B=3
A/B=333...2
Input a instruction:_
```



```
ASM3 [正在运行] - Oracle VM VirtualBox
控制 视图 设备 帮助
Input a instruction:8
A=99999999
B=2
A/B=49999999...1
Input a instruction:_
```

除法溢出时



ASM3 [正在运行] - Oracle VM VirtualBox

控制 视图 设备 帮助

```
Input a instruction:8
A=1001
B=3
A/B=333...2

Input a instruction:8
A=100
B=0
#Divide Overflow!!!!_
```



ASM3 [正在运行] - Oracle VM VirtualBox

控制 视图 设备 帮助

```
Input a instruction:8
A=99999999
B=2
A/B=49999999...1

Input a instruction:8
A=6
B=0
#Divide Overflow!!!
Input a instruction:_
```

#### 4. INT09H-KeyBoard Handler

输入“wuwurqerq”，显示自己的学号



ASM3 [正在运行] - Oracle VM VirtualBox

控制 视图 设备 帮助

```
Input a instruction:9
Here is the new INT 09H keyboard handler
Q=0 W=2 ... P=9
161630230_
```

#### 5. INT1CH-System Real Time

实际上此处可实时变化，图片显示不出效果。空格退出程序。



ASM3 [正在运行] - Oracle VM VirtualBox

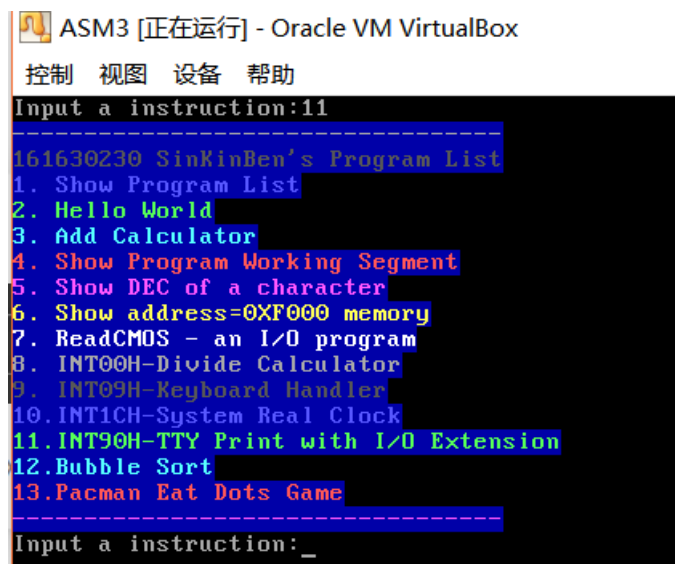
控制 视图 设备 帮助

```
Input a instruction:10
20:40:12
_
```

#### 6. TTY Print with I/O Extension

在自定义的 INT 90H 中断下输出的程序列表，该程序自定义了带属性输出 TTY 方式，并实现自动后移光标位置。

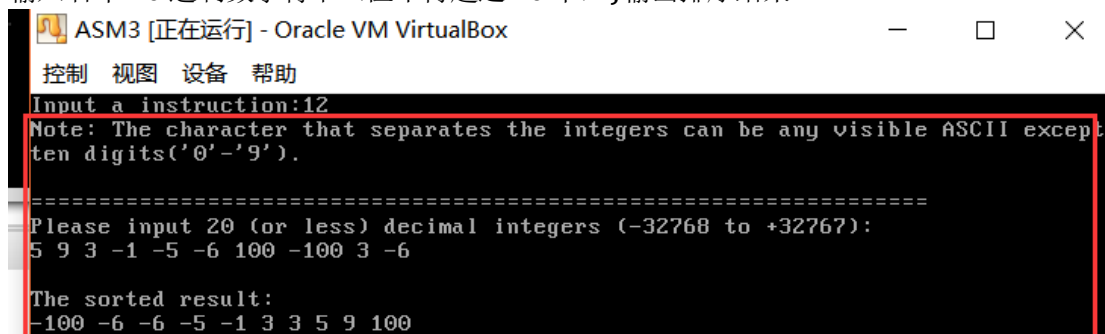




```
ASM3 [正在运行] - Oracle VM VirtualBox
控制 视图 设备 帮助
Input a instruction:11
-----
161630230 SinKinBen's Program List
1. Show Program List
2. Hello World
3. Add Calculator
4. Show Program Working Segment
5. Show DEC of a character
6. Show address=0XF000 memory
7. ReadCMOS - an I/O program
8. INT00H-Divide Calculator
9. INT09H-KeyBoard Handler
10. INT1CH-System Real Clock
11. INT90H-TTY Print with I/O Extension
12. Bubble Sort
13. Pacman Eat Dots Game
-----
Input a instruction:_
```

### 7. Bubble Sort

输入若干 10 进制数字字符串（但不得超过 20 个），输出排序结果



```
ASM3 [正在运行] - Oracle VM VirtualBox
控制 视图 设备 帮助
Input a instruction:12
Note: The character that separates the integers can be any visible ASCII except
ten digits('0'-'9').
=====
Please input 20 (or less) decimal integers (-32768 to +32767):
5 9 3 -1 -5 -6 100 -100 3 -6

The sorted result:
-100 -6 -6 -5 -1 3 3 5 9 100
```

### 8. Pacman Eat Dots Game


操作对象:  吃豆人 Pacman

操作方式: 键盘输入, a 向左, d 向右, w 向上, s 向下。

目标: 吃完所有豆子则胜利。

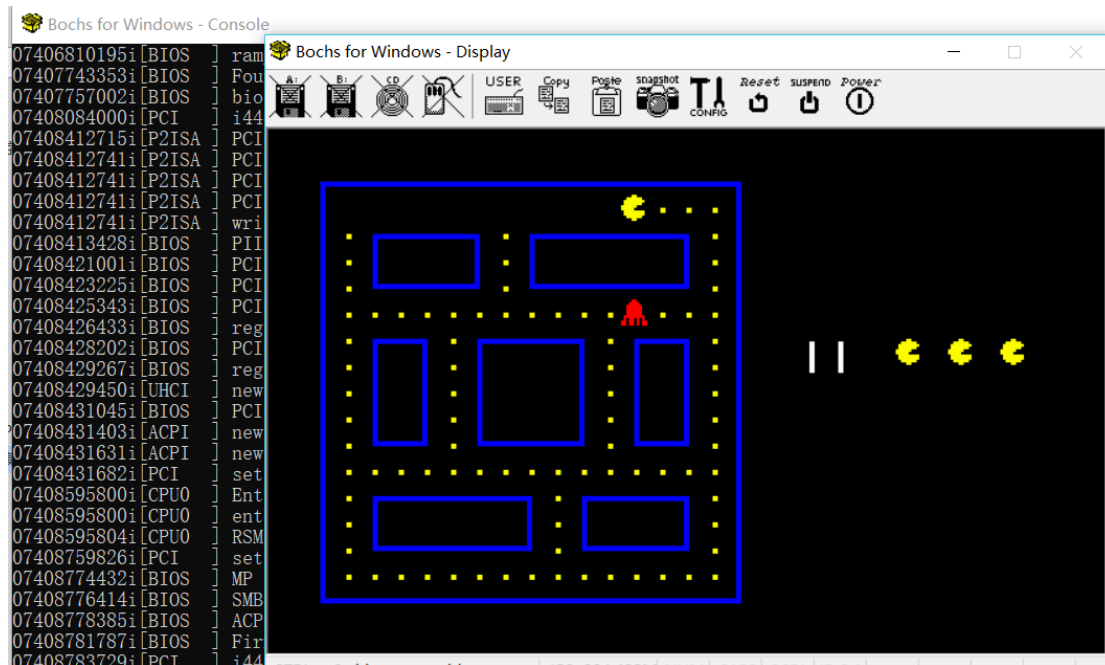
分数: , 显示吃豆人吃了几颗豆子。

生命: 

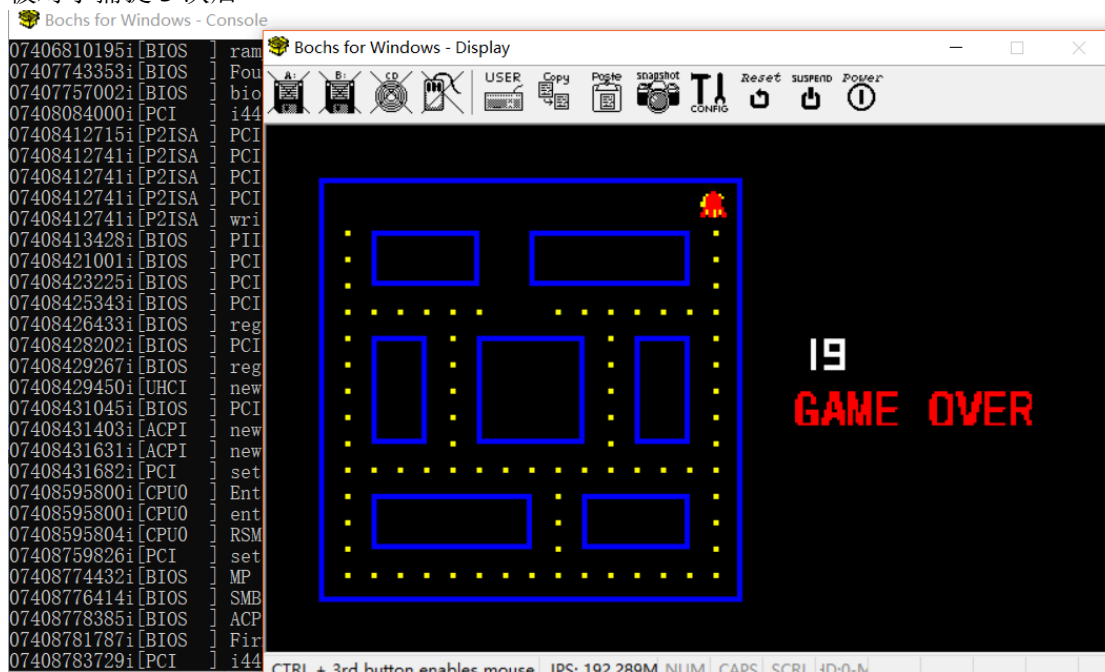
对手: 导弹 , 随着 Pacman 的移动而移动, Pacman 被捕获一次则生命减一, 失去 3 次生命则失败, 游戏结束。

## 3.3 Bochs 环境下实现 Pacman

W 键为  $\uparrow$ , S 为  $\downarrow$ , A 为  $\leftarrow$ , D 为  $\rightarrow$ 。该功能是通过实现自定义的中断程序实现的。(即改进程序 9. Int09H-keyboard, 使处理器只响应键盘 W,S,A,D 这 4 个键的中断, 并将其映射到功能键  $\uparrow$ ,  $\downarrow$ ,  $\leftarrow$ ,  $\rightarrow$ )



被对手捕捉 3 次后

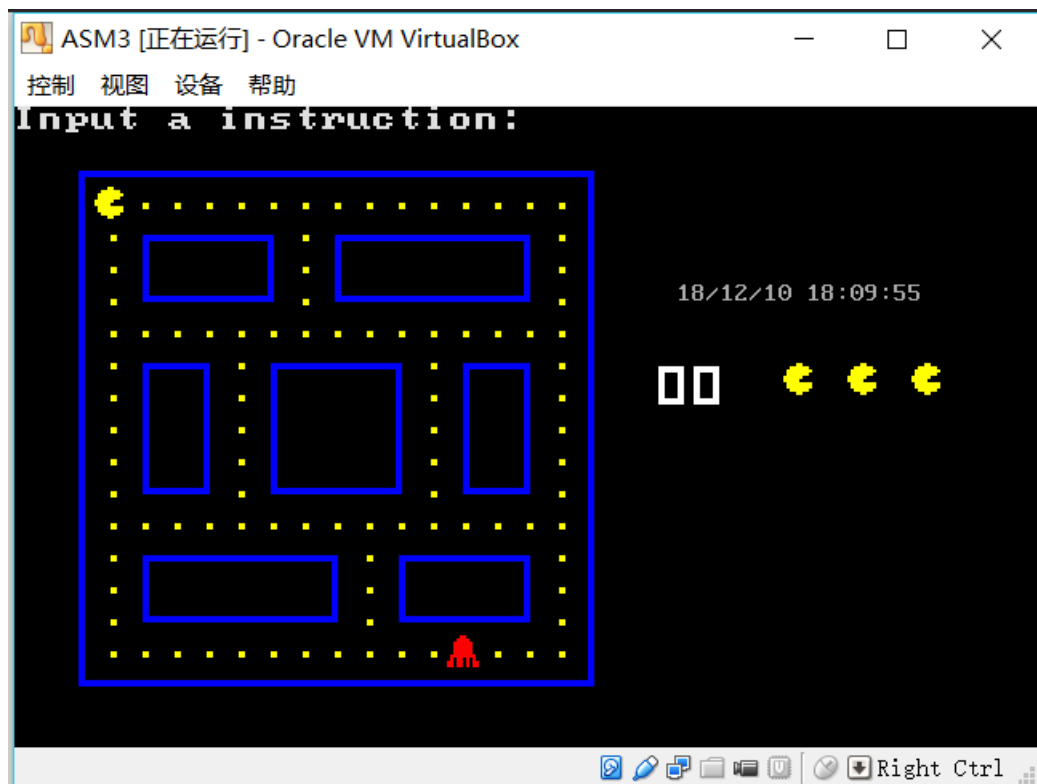


### 3.4 VirtualBox 环境下集成的 Pacman

程序进入 myloader 后，输入“13”即可运行。

右侧加入了实时时钟的显示。（基于自定义中断服务程序 system real clock 实现）

该游戏的实现请参考 4.3 小节。



## 四、实现思路

### 4.1 整体架构

myloader 的实现主体上的设计是模仿课本的例子 DP77.asm 实现的，在模仿的基础上加入了自己的个性化功能。

执行步骤大致如下：

0. 启动虚拟机，BIOS 启动，加载 0 扇区，控制权转交给 Master boot record，在这里即是 myloader。

1. 设置堆栈（SS 和 SP 寄存器），设置 myloader 本身的代码段和数据段（即 DS 和 CS 寄存器）。（07C00 -> 00000 是堆栈空间）

2. 获取用户输入的命令（实际上每个命令都对应一个程序）

3. 确定命令对应的程序

4. 读取程序首个扇区

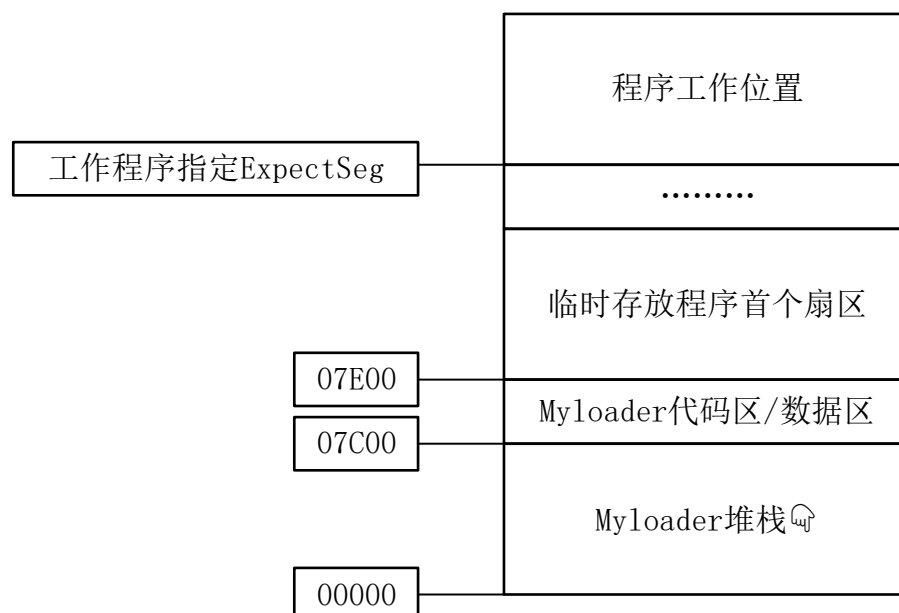
5. 程序文件头验证。

6. 把程序加载到内存起始位置（即 ExpertedSeg）

7. 如果大于 1 个扇区，读取全部，最后跳转执行。

（更多的实现细节可以参考 0-myloader.asm 文件）

程序的内存布局大致如下：



## 4.2 中断服务程序

### 1. Int00H-Devide Calculator

- 功能描述：作为一个简单的除法程序，输入两个十进制数 **A,B**（不超过  $2^{32}$ ）。若无除法溢出，则输出 **A/B** 的商和余数；若除法溢出（例如 **B=0**），则调用自定义的 **INT 00H** 中断程序，提示除法溢出。
- 大致框架如 1.5 节“中断服务程序设计”所述。
- 在此基础上要实现以下几个子程序，都是一些常见的程序。

子程序	功能描述
GetStr	获取一个字符串
DSTOB	10 进制字符串转换为二进制数，存放在 <b>EAX</b> 中
BIN2DEC	将 <b>EAX</b> 转换为 10 进制字符串输出
PutStr	<b>INT 10H</b> BIOS 调用，输出结果字符串

### 2. Int09H-KeyBoard Handler

该程序的难点在于管理一个**环形队列**数据结构作为键盘扫描码的缓冲区。根据课本的样例程序的提示，键盘输入缓冲区位于 BIOS 的数据区，数据结构如下：

```

BUFF_HEAD  DW      001EH  ;0040:001AH

```

```
BUFF_TAIL DW 001EH ;0040:001CH
```

```
KB_BUFFER RESW 16
```

;16 个字（32 字节），每个字用于存放一个按键的扫描码和 ASCII 码

此外，管理队列的程序课本的样例程序已给出，可仿照实现。

```
;-----
Enqueue:                                ;把扫描码和ASCII码存入键盘缓冲区
    PUSH DS                            ;保护DS
    MOV BX, 40H
    MOV DS, BX                        ;DS=0040H
    MOV BX, [001CH]                   ;取队列的尾指针
    MOV SI, BX                        ;SI=队列尾指针
    ADD SI, 2                          ;SI=下一个可能位置
    CMP SI, 003EH                     ;超出缓冲区界吗？
    JB .LAB1                          ;没有，转
    MOV SI, 001EH                     ;是的，循环到缓冲区头部
.LAB1:
    CMP SI, [001AH]                   ;与队列头指针比较
    JZ .LAB2                          ;相等表示，队列已经满
    MOV [BX], AX                      ;把扫描码和ASCII码填入队列
    MOV [001CH], SI                  ;保存队列尾指针
.LAB2:
    POP DS                            ;恢复DS
    RET                              ;返回
;-----
```

### 3. Int1CH-System Real Clock

本程序基于程序 7.ReadCMOS 实现，把 ReadCMOS 获取时间程序改造为一个子程序（开辟了 hour, minute, second 三个数据项存放时分秒）。

开辟字节数据 count 作为计数器。令 count=18，通过循环和 NOP 指令进行延时操作，每当 count=0 时，刷新显示页，读取一次时间并显示之，重置 count=18。代码如下：

```

;-----
;system real time int 1CH
new_handler_1CH:
    DEC     BYTE [CS:count]
    JZ      PrintTime
    IRET
;
PrintTime:
    MOV     BYTE [CS:count], 18
    ;
    STI
    PUSHA
    CALL    get_time           ;获取当前时间
    CALL    EchoTime          ;显示当前时间
    POPA
    IRET
;-----

```

### 4.3 吃豆豆小游戏

本程序集成了 3 个自定义的中断服务程序，如下表：

中断类型号	程序	描述
09H	newKeyBoard	自定义键盘中断，使得在游戏进行时，只有 W,A,S,D 在起作用，负责 Pacman 的上下左右移动。基于程序 9.Int09H-keyboard 实现。
1CH	System real time	在游戏窗口中，显示系统实时时间。基于程序 10.System real time 实现。
90H	IOExtension	TTY 方式+彩色输出。基于程序 11. Int90H-IOExtension 实现。

实现的子程序的功能：

子程序	功能描述
Drawwall	画地图的墙壁
ChangePos	改变吃豆人的位置，是小游戏的核心程序
Drawbeans	画豆子
DrawPacman	画吃豆人
DrawMissile	画对手导弹

DrawNumber	画得分的数字
Result	处理失去生命的情况；并输出比赛结果 WIN/Failure
DrawLife	画出 Pacman 还剩下几个生命

1. 首先，我们让“裸机”进入图形显示模式（通过 BIOS 调用实现），代码如下：

;进入图形模式

```
mov al,0x13
```

```
mov ah,0x00
```

```
int 0x10
```

2. 然后设置颜色表，这一部分是一个简单的机械劳动，查阅 RGB 颜色表对应的 BIOS 调用参数即可实现，通过 OUT 指令写到对应的 IO 端口。例如设置黑色如下：

;-----设置颜色表-----

;黑色

```
mov al,1
```

```
mov dx,0x3c8
```

```
out dx,al
```

;rgb 颜色赋值

```
mov al,0;红
```

```
mov dx,0x3c9
```

```
out dx,al
```

```
mov al,0;绿
```

```
out dx,al
```

```
mov al,0;蓝
```

```
out dx,al
```

3. 通过下面所述的方法描绘出各种游戏元素。

各种游戏元素的实现如下：





串存到 DEC\_STR 的内存缓冲区。(通过 INT 21H, AH=0AH 实现)

2. 十进制字符串转换到二进制数据: 扫描字符串, 以空格为分隔符进行判断, 识别出每个数字, 转换为二进制数, 存放到 DEC\_NUM 的内存缓冲区。(通过 ASCTODEC 子程序实现)。
3. 对 DEC\_NUM 缓冲区中的数组进行冒泡排序 (在原地进行排序)。(通过 BubbleSort 子程序实现)
4. 将 DEC\_NUM 中的数组转换为十进制字符串输出。(通过子程序 DECTOASC 实现)。

上述的几个子程序都是平时作业常用的, 因此实现起来都没难度, 就不一一细说了。

程序的运行效果如下:

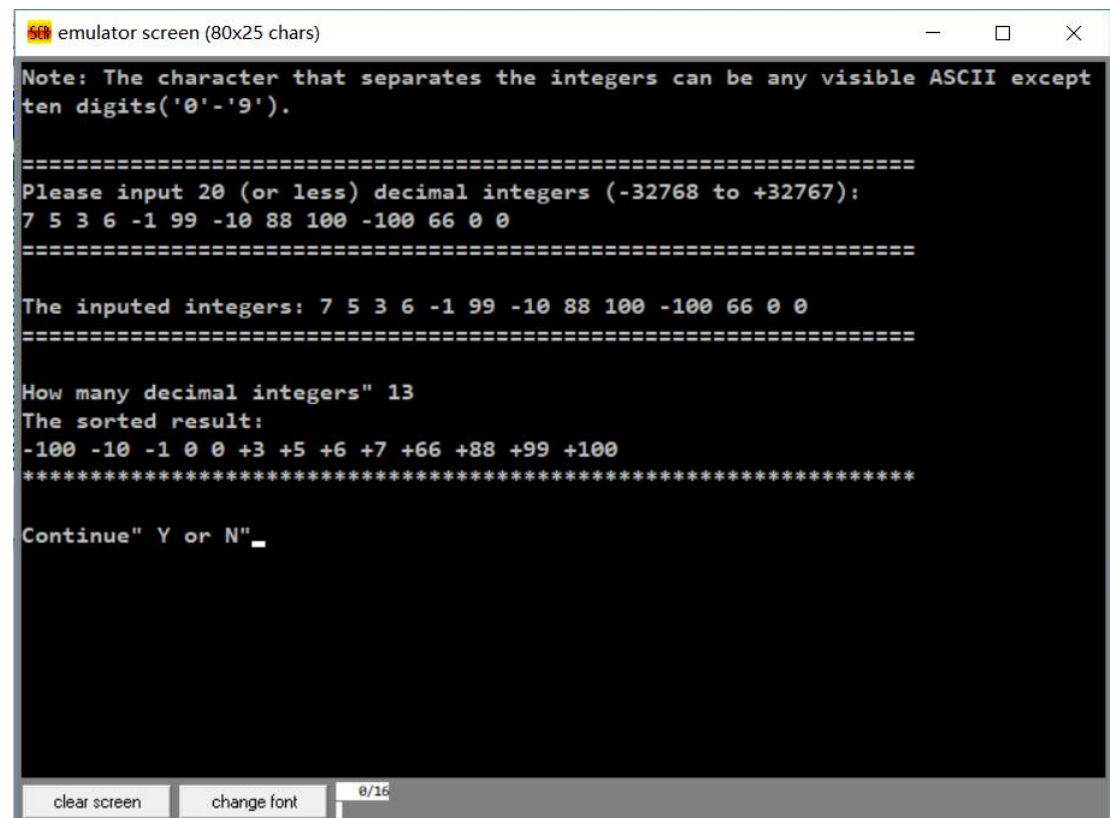
第一步: 输入若干个数字(个数自由, 但不得超过 20 个, 以空格分开)。

第二步: 输出分割后的字符串, 确定 ASCTODEC 程序的正确性。

第三步: 输出排序后的结果 (正数带 “+” 号, 负数带 “-” 号)。

第四步: 允许进行下一次的输入和排序, 输入 N/n 结束程序。

复杂数据测试:



```
emulator screen (80x25 chars)
Note: The character that separates the integers can be any visible ASCII except
ten digits('0'-'9').

=====
Please input 20 (or less) decimal integers (-32768 to +32767):
7 5 3 6 -1 99 -10 88 100 -100 66 0 0
=====

The inputed integers: 7 5 3 6 -1 99 -10 88 100 -100 66 0 0
=====

How many decimal integers" 13
The sorted result:
-100 -10 -1 0 0 +3 +5 +6 +7 +66 +88 +99 +100
*****

Continue" Y or N" _
```

clear screen    change font    0/16

```
emulator screen (80x25 chars)

The inputed integers: 7 5 3 6 -1 99 -10 88 100 -100 66 0 0
=====

How many decimal integers" 13
The sorted result:
-100 -10 -1 0 0 +3 +5 +6 +7 +66 +88 +99 +100
*****

Continue" Y or N"y
=====
Please input 20 (or less) decimal integers (-32768 to +32767):
1 10000 2000 336 651 102 -8 -5 0 3 6 4 111 -999 55 -999
=====

The inputed integers: 1 10000 2000 336 651 102 -8 -5 0 3 6 4 111 -999 55 -999
=====

How many decimal integers" 16
The sorted result:
-999 -999 -8 -5 0 +1 +3 +4 +6 +55 +102 +111 +336 +651 +2000 +10000
*****

Continue" Y or N"
```

特殊数据测试:

```
emulator screen (80x25 chars)

Note: The character that separates the integers can be any visible ASCII except
ten digits('0'-'9').

=====
Please input 20 (or less) decimal integers (-32768 to +32767):
1 1 1
=====

The inputed integers: 1 1 1
=====

How many decimal integers" 3
The sorted result:
+1 +1 +1
*****

Continue" Y or N"
```

## 参考文献

- [1] 杨季文, 新概念汇编语言 [M], 北京: 清华大学出版社, 2017
- [2] 王爽, 汇编语言 (第二版) [M], 北京: 清华大学出版社, 2004
- [3] 于渊, 自己动手写操作系统[M], 北京: 电子工业出版社, 2008
- [4] <https://blog.csdn.net/xiaominthere/article/details/17118829>
- [5] <https://blog.csdn.net/piaopiaopiaopiaopiao/article/details/9735633>
- [6] <https://wenku.baidu.com/view/9be1ae2cabea998fcc22bcd126fff705cc175c1d.html> (BIOS 调用大全, 包含入口和出口参数)
- [7] <https://www.supfree.net/search.asp?id=6386> (键盘按键的扫描码和 ASCII 码)