1.问题清单笔记

- 请提交运行截图和代码

2 问题清单

在整个实验过程中，无论是编程还是查资料，请同学们注意思考以下问题，助教检查时会从中随机抽取数个题目进行提问，根据现场作答给出分数。请注意，我们鼓励自己思考和动手实验，如果能够提供自己的思考结果并辅助以相应的实验结果进行说明，在分数评定上会酌情考虑。

1. 请简述 80×86 系列的发展历史
2. 说明小端和大端的区别，并说明 80×86 系列采用了哪种方式?
3. 8086 有哪五类寄存器，请分别举例说明其作用?
4. 什么是寻址？立即寻址和直接寻址的区别是什么?
5. 请举例说明寄存器间接寻址、寄存器相对寻址、基址加变址寻址、相对基址加变址寻址四种方式的区别
6. 请分别简述 MOV 指令和 LEA 指令的用法和作用?
7. 请说出主程序与子程序之间至少三种参数传递方式
8. 如何处理输入和输出，代码中哪里体现出来?
9. 有哪些段寄存器
10. 通过什么寄存器保存前一次的运算结果，在代码中哪里体现出来。
11. 解释 boot.asm 文件中，`org 0700h` 的作用
12. boot.bin 应该放在软盘的哪一个扇区？为什么?
13. loader 的作用有哪些?
14. 解释 NASM 语言中 `[ ]` 的作用
15. 解释语句 `times 510-($-$$) db 0`，为什么是 510? `$` 和 `$$` 分别表示什么?
16. 解释配置文件 bochsrc 文件中如下参数的含义

```
megs:32
display_library: sdl
floppya: 1_44=a.img, status=inserted
boot: floppy
```

1.1 80x86系列发展史

1978年6月，intel推出第一款16位处理器8086,20位地址线；

1982年发布80286，主频提高到12MHz

1985年发布80386，处理器变为32位，地址线也扩展到32位；

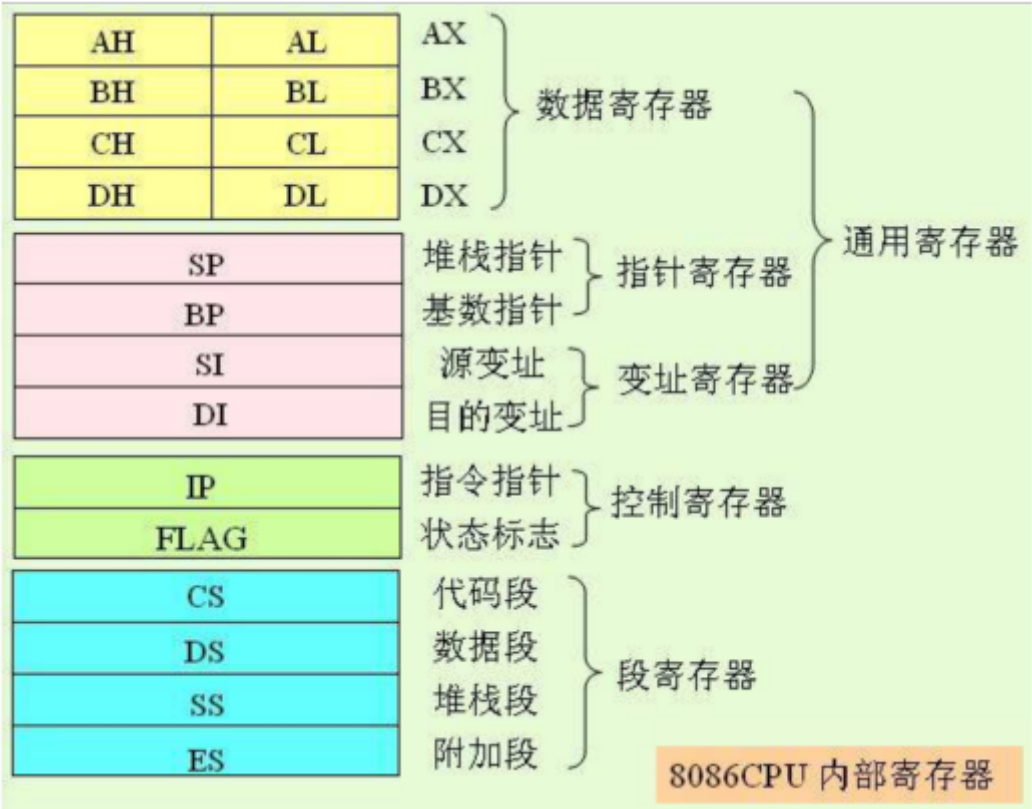
89年发布486,93年发布586,并命名为奔腾；

1.2 大端还是小端?

8086系列采用的是小端存储方式;

大/小端存储指的是: 对于内存中的低号地址，是先把数据的高位存进去，还是低位存进去; 先把高位存进去，那就是大端存储，先把低位存进去，就是小端存储;

1.3 8086寄存器有哪5类？举例并说明作用;



总共是3大类，5小类:

a.段寄存器

CS 代码段寄存器：存储程序代码区的开始地址;

b.控制寄存器

FLAG 状态标志寄存器的不同位有不同意义; 包括零标志、进位标志、溢出标志等

c.通用寄存器

c1.数据寄存器

EAX:存储变量/数据/地址等等

c2.

指针寄存器

SP Stack Pointer 配合Stack Segment寄存器,共同指向目前的栈的地址;

BP Base Pointer 基址指针寄存器;可以当做相对基址;

c1.

变址寄存器

SI 源变址寄存器，存放相对于数据段寄存器的变址指针

DI Destination 目的变址寄存器，存放相对于扩展段寄存器的变址指针;

1.4 寻址？立即寻址和直接寻址

寻址：找到操作数的地址，从而取得操作数

立即寻址：命令后跟随的参数是操作数本身，实际上没有经过寻址的过程；

直接寻址：命令后跟随的参数是操作数的地址; 通过取址可以获得操作数;

1.5

寄存器间接寻址：指向操作数的地址在一个寄存器中;

寄存器相对寻址：某个寄存器中的地址加上偏移量，就是指向操作数的地址;

基址加变址：基址寄存器的内容（BX\BP)加上变址寄存器(SI\DI)的内容，才等于操作数的地址

相对基址加变址：基址寄存器和变址寄存器的和，再加上一个偏移量，才等于操作数的地址

1.6 mov指令和lea指令的用法和作用

mov指令是一条赋值指令，它的特点是赋值的2边大小应该一样；可以在寄存器与寄存器间，寄存器与内存，寄存器与立即数，内存与立即数之间使用。

比如都是1个字节的mov al,cl；立即数是最大的，32位或者64位；

mov 源，目的地

lea指令将内存里某个单元的4位16进制偏移地址送到指定的寄存器。 LEA reg16,mem

1.7 主程序和子程序间传参的方式

1.寄存器传参

缺点：寄存器有限，可传递的参数个数有限

2.约定的地址传参

3.利用堆栈传参（常用）

1.8 如何处理输入和输出，代码体现

系统调用 0(64位read)、1（64位write）

看代码

1.9 有哪些段寄存器

CS 代码段寄存器：存储代码区的开始位置

DS 数据段寄存器：存储数据区的开始位置

SS 堆栈段寄存器：存储堆栈区的开始位置

ES 扩展段寄存器

1.10 哪个寄存器保存前一个的运算结果；在代码哪里表现出来

一般在64位系统下，约定用rax和rdx存储函数的返回值；

代码的体现在

1.11 解释boot.asm中，org 07c00h作用

1.BIOS约定软盘内容放在07c00h位置，并不是由这行代码决定的。

2.**org是伪指令**，是告诉汇编器，这段代码需要放到07c00h这个地方。以后遇到绝对寻址的指令，地址就是07c00加上相对地址；

12 boot.bin应该放在哪一个扇区?

放在软盘的0面0磁道1扇区；

BIOS程序是厂家写好的，开机时会自动运行，他检查软盘的0面0磁道1扇区，如果扇区以0xaa55结束，认定为引导扇区，将512字节数据加载到07c处，然后跳转到这个地方执行代码。

1.13 loader作用是什么?

loader是一个开机的引导程序，他可以启动保护模式、开启分页机制，然后将内核加载到内存中，并且运行内核。

1.14 解释NASM中[]的作用；

这相当于是解引用的符号，一般前面还要加BYTE\WORD\DWORD\QWORD等限定词，用以说明怎样解释“类型”；解释多少个字节的内容

1.15 解释$和$$的作用

一个扇区512个字节，结束符aa55占了2个字节；

将剩下的扇区空间用0填充，直到将510个字节填满；

一个$代表本行代码所在地址；

$$代表当前代码段所在地址；

$-$$代表已经有多少个字节是占用的；

1.16 解释bochs参数

megs:32 表示虚拟机的内存大小

sdl表示用sdl库进行显示

floppya什么映像会被使用

boot:floppy驱动盘是软盘；

1.17 编译命令

nasm -f elf64 BigNum.asm -o BigNum.o

nasm BigNum.o -o BigNum -no-pie

./BigNum