**问题清单:**

⾏号 代码

1 mov al, [L1]

2 mov eax, L1

3 mov [L1], ah

4 mov eax, [L6]

5 add eax, [L6]

6 add [L6], eax

7 mov al, [L6]

1. boot.asm⽂件中，org 0700h的作⽤

使汇编程序依据的是相对地址与内存绝对地址吻合

1. 为什么要把boot.bin放在第⼀个扇区？直接复制为什么不⾏？

将以0xaa55结束的boot.bin放在第一个扇区，使其作为引导扇区，借由BIOS加载到内存0x7c00处并执行；普通的读写操作（mv，rm，cp）是基于文件系统这一逻辑概念的，而引导扇区属于物理概念，无法直接复制。

1. loader的作⽤有哪些?

从kernel.bin中提取内核代码加载到内存

从实模式跳入保护模式

启动内存分页

跳转到内核开始地址，执行内核

1. L1，L6各标识了⼀个字节（8bit）的数据，eax是⼀个32位寄存器，说明上⾯每⾏代码的意思。

⾏号 代码

1 mov al, [L1] 将L1标识的字节数据加载到al中

2 mov eax, L1 将L1的逻辑地址加载到eax中

3 mov [L1], ah 将ah中的内容加载到L1标志处

4 mov eax, [L6] 将L6标志的数据加载到eax中

5 add eax, [L6] 将eax中的数据加上L6标识的数据

6 add [L6], eax 将L6标识的数据加上eax中的值

7 mov al, [L6] 将L6标识的数据加载到al

1. times 510-($-$$) db 0中的数字为什么是510？$和$$分别表⽰什么？不⽤times指令怎么写（等价命令）？

引导扇区总共512个字节，减去结尾的0xaa55正好是510个字节

$表示当前行被汇编后的地址，$$表示一个节（section）的开始处被汇编后的地址

等价命令：resb 510-($-$$)

1. 解释L10 db "w", "o", "r", "d", 0这条语句的意义，并且说明数字0的作⽤。

在以L10为首地址标识的内存中依次写入字节"w", "o", "r", "d", 0

数字0作为字符串结束标识，可被Write Sting系统调用识别

1. L1 db 0 L2 dw 1000中的L1、L2是连续存储的吗？即是否L2就存储在L1之后？

L1、L2是连续存储的，L2就存储在L1之后

1. 要是不知道L6标识的是多⼤的数据，下⾯这句话对不对？mov [L6], 1

不对，会报错：operation size not specified

1. 如何处理输⼊输出？在代码中哪⾥体现出来？

输入输出分别通过第3、4号系统调用执行，处理部分结合代码（Fibonacci.asm）

1. 通过什么来保存前⼀次的运算结果？在代码中哪⾥体现出来？

通过在内存中开辟的空间（标记为n、n\_1、n\_2）保存运算结果

1. 有哪些段寄存器？

CS:代码段寄存器

DS:数据段寄存器

ES:附加段寄存器

SS:堆栈段寄存器

12.8086/8088存储单元的物理地址⻓，CPU总线的数量，可以直接寻址的物理地址空间。

8086/8088存储单元的物理地址⻓20位，范围是00000H至FFFFFH

CPU总线的数量：20根地址线，16根数据线

可以直接寻址的物理地址空间是2的20次方，即1M字节

13.如何根据逻辑地址计算物理地址？

存储单元的逻辑地址表示为段值：偏移，

物理地址=段值\*16（左移4位）+偏移

14.寄存器的寻址⽅式（知道如何计算）

**立即寻址方式**

MOV AX, 1234H

立即数可以是8位，也可以是16位

高高低低

**寄存器寻址方式**

MOV SI,AX

MOV AL,DH

16位操作数寄存器是AX,BX,CX,DX,SI,DI,SP,BP

8位操作数寄存器是AL,AH,BL,BH,CL,CH,DL,DH

**直接寻址方式**

MOV AX,[1234H]

设寄存器DS的内容是5000H

MOV ES:[5678H],BL

引用的段寄存器是ES

直接寻址的地址要放在方括号中

**寄存器间接寻址方式**

MOV AX,[SI]

(DS)=5000H,(SI)=1234H

MOV DL,CS:[BX]

引用的段寄存器是CS

MOV [BP],CX

缺省的段寄存器是SS

**寄存器相对寻址方式**

(DS)=5000H,(DI)=3678H

MOV AX,[DI+1223H]

MOV BX,[BP-4]

MOV ES:[BX+5],AL

以下两种等价

MOV AX,[SI+3]

MOV AX,3[SI]

指令中给定的8位和16位位移量采用补码表示，在计算有效地址时，如果位移量8位带符号拓展为16位 如果有效地址超过FFFFH，取其64K的模

**基址加变址寻址方式**

(DS)=5000H,(BX)=1223H,(DI)=54H

MOV AX,[BX+DI]

MOV AX,ES:[BX+SI]

MOV DS:[BP+SI],AL

以下两种等价

MOV AX,[BX+DI]

MOV AX,[DI][BX]

**相对基址加变址寻址方式**

MOV AX,[BX+DI-2]

(DS)=5000H,(BX)=1223H,(DI)=54H

以下四种等价

MOV AX,[BX+DI+1234H]

MOV AX,1234H[BX+DI]

MOV AX,1234H[BX][DI]

MOV AX,1234H[DI][BX]

15.⼏个常⽤指令的作⽤（如MOV，LEA等）

MOV:传送数据

把数据从源操作SRC数送至目的操作数DST ，内部寄存器之间的数据传送，立即数送至通用寄存器或存储单元，寄存器与存储器的数据传送

LEA:传送地址

把操作数OPRD的有效地址传送到操作数REG

操作数OPRD必须是一个存储器操作数

16.主程序与⼦程序的⼏种参数传递⽅式

**利用寄存器传递参数**

• 利用寄存器传递参数就是把参数放到约定的寄存器中。

• 这种方法的优点是实现简单和调用方便。

• 但是由于寄存器的个数是有限的，且寄存器往往还要存放其他数据，所以只适用于要传递的参数较少的情况

**利用约定存储单元传递参数**

• 在传递参数较多的情况下，可利用约定的内存变量来传递参数。

• 这种方法的优点是子程序要处理的数据或送出的结果都有独立的存储单元，编写程序时不易出错。

• 但是这种方法要占用一定的存储单元。

• 利用约定的存储单元传递参数，通用性较差。为了传递较多的参数，又要保持良好的通用性，通常把参数组织成一张参数表，存放在某个存储区，然后把这个存储区的首地址传送给子程序。（既可以利用寄存器传递首地址，也可利用堆栈方法传递首地址）

**利用堆栈传递参数**

• 如果使用堆栈传递参数，那么主程序在调用子程序之前，把需要传递的参数依次压入堆栈，子程序从堆栈中取入口参数；如果使用堆栈传递出口参数，那么子程序在返回前，把需要返回的参数存入堆栈，主程序在堆栈中取出口参数。

• 利用堆栈传递参数可以不占用寄存器，也无需使用额外的存储单元。

• 但是由于参数和子程序的返回地址混杂在一起，有时还要考虑保护寄存器，所以较为复杂。 • 通常堆栈传递入口参数，而利用寄存器传递出口参数。

**利用CALL后续区传递参数**

• CALL后续区是指位于CALL指令后的存储区域。

• 主程序在调用子程序之前，把入口参数存入CALL指令后的存储单元中，子程序根据保存在堆栈中的返回地址找到入口参数——这种传递参数的方法称为 CALL后续区传递参数法。 • 利用CALL后续区传递参数的子程序必须修改返回地址。由于这种方法把数据和代码混在一起，所以在x86系列汇编语言程序中使用不多