1、CPU 是哪 2 个部件集成到一个芯片

① 运算器 ALU；② 控制器 CU

2、计算机系统包括哪 2 个部分

① **硬件：**计算机的实体，如主机、外设

② **软件：**由具有各类特殊功能的信息（程序）组成

3、计算机软件包括哪 2 种类型

① **系统软件：**用来管理整个计算机系统

② **应用软件：**按任务需要编制成的各种程序

4、CPU 由哪几部分组成

运算器、控制器、寄存器和内部总线

5、8086 的有哪几个通用寄存器?哪些提供了字节访问

① AX；② BX；③ CX；④ DX

提供字节访问：

① AH，AL；② BH，BL；③ CH，CL；④ DH，DL

6、标志寄存器各标志位的含义？IP 和 SP 的含义

I、各标志位的含义：

① **ZF**（zero flag）：零标志

ZF = 0：结果不为 0

= 1：结果为

② **CF（**carry flag）：进位标志位

CF = 0：没有进位或借位

= 1：有借位

③ **AF (auxiliary flag)：辅**助进位标志

AF = 0：有进位或借位

= 1：有进位或借位

④ **PF（**parity flag）：奇偶标志位

PF = 0：为奇数

= 1：为 0 数

⑤ **SF**（sample flag）：符号标志位

SF = 0：结果为正

= 1：结果为负

⑥ **OF（**overflow flag）：溢出标志位

OF = 0：结果无溢出

= 1：结果有溢出

⑦ **IF（**interrupt flag）：中断标志位

IF = 0：关中断

IF = 1：开中断

⑧ **TF**（track flag）：跟踪标志位

TF = 0：禁止单步中断

= 1：允许单步中断

⑨ **DF**（direction flag）：方向标志位

DF = 0：地址指针递增修改

= 1：地址指针递减修改

II、IP 和 SP 含义：

① **IP：**16 位指令指针寄存器，存放下一条要取的指令的偏移地址，具有自动加一功能

② **SP：**16 位堆栈指针寄存器

7、8086 有哪几个段寄存器

① **CS**（code segment）：代码段寄存器
② **DS**（data segment）：数据段寄存器
③ **ES**（extra segment）：附加段寄存器
④ **SS**（stack segment）：堆栈段寄存器

8、了解寄存器和存储器的不同

① **寄存器**存在于 CPU 中，速度很快，数目有限；物理结构上是 D 触发器
② **存储器**在 CPU 外，是用来存放数据、程序以及运算中间结果的部件

9、8086 的一个字(16 位)怎样存入存储器的

低 8 位存入存储器的低地址单元，**高 8 位**存入存储器的高地址单元

10、8086 的使用存储器地址访问数据时，能根据地址确定数据的大小吗

不能

11、8086 的地址空间是多大

8086 共 20 位地址总线，地址空间为 2^{20}B

12、1 个段最大最小都是多大，段首地址的特点

① 段最小为 0KB；由于 IP（指令指针寄存器）长度为 16 位，所以最大为 2^{16}KB
② 段首地址要求低 4 位全为 0

13、一个物理地址由段地址和偏移地址组成，这种表示是唯一的吗

不唯一

14、学会段地址和偏移地址合成物理地址

物理地址 = 段地址 * 16 + 偏移地址

15、8086 实模式下内存最多多大

2^{20}KB

寻址方式与指令

1、 能识别各种寻址方式，尤其是存储器寻址

操作码	操作数

操作码：指令操作类型；**操作数：**指令所需操作数或操作数地址

I、立即寻址方式

操作数通过指令直接给出，操作数与操作码一起存入代码段中

如 mov ax, 5

II、寄存器寻址方式

操作数是寄存器的值，指令中直接使用寄存器名，包括 8 位或 16 位通用寄存器和段寄存器

如 mov ax, bx （AX）← （BX）

III、存储器寻址

① 有效地址组成：

i、偏移量：存放在指令中的一个 8 位、16 位或 32 位的数，是地址，不是立即数

ii、基址：存放在基址寄存器（BP base point）中的内容，常用于指向数组或字符串首地址

iii、变址：存放在变址寄存器（SI source index 源变址寄存器；DI destination index 目的变址寄存器）中的内容，常用于访问数组或字符串中的某个字符

② 各访存类型下默认段的选择

以下三种情况，不允许使用段跨越前缀

i、串处理指令目的串必须在 ES

ii、PUSH 指令的目的和 POP 的源必须在 SS

iii、指令必须存放在 CS

访存类型	所用段及段寄存器	缺省选择规则
取指	代码段 CS	用于取指
堆栈	堆栈段 SS	所有进栈、出栈，任何使用 BP 作为基址寄存器的访存
局部数据	数据段 DS	除相对于堆栈和串处理指令目的串之外的所有数据访问
目的串	附加段 ES	串处理指令的目的串

① 直接寻址方式

位移量的值为操作数的有效地址，存放在代码段操作码之后。操作数地址为：DS：偏移地址或 ES：偏移地址。隐含指定 DS

mov ax, [3000h]

mov ax,. [ary] / ary

② 寄存器间接寻址方式

操作数的有效地址位于 BX、BP、SI 或 DI 中

mov ax, [bx] / [bp] / [si] / [di]

物理地址 = (DS) * 16 + (BX) / (SI) / (Di)

= (SS) * 16 + (BP)

③ 寄存器相对寻址方式

寄存器内容与位移量之和作为操作数所在单元的有效地址

mov ax, [bx + count] / count[bx]

mov ax, 1[bx]

物理地址 = (DS) * 16 + (BX) / (SI) / (Di) + DISP8 / DISP16

= (SS) * 16 + (BP) + DISP8 / DISP16

④ 基址变址寻址方式

基址寄存器（BX 或 BP）与变址寄存器（SI 或 DI）之和作为操作数所在存储单元的有效地址

以 BP 做基址寄存器取得是 SS 中的内容

mov ax, [bp + si] / [bp][si] / [si][bp] / [si + bp]

mov ax, [bp + di]

mov ax, [bx + si]

mov ax, [bx + di]

⑤ 相对基址变址寻址方式

基址寄存器（BX 或 BP）与变址寄存器（SI 或 DI）内容之和再加上 8 位或 16 位位移量之和作为操作数所在单元的有效地址

2、理解立即寻址方式

立即寻址方式，在指令中直接给出操作数，操作数与操作码一起放入代码段中

3、寄存器寻址和寄存器间接寻址的区别

寄存器寻址操作数是寄存器的值

寄存器间接寻址是内存寻址，寄存器中存放有效地址

4、8086 存储器寻址的有效地址如何构成

① **直接寻址：**偏移地址（EA）由指令直接给出

② **寄存器间接寻址：**（EA）=（BX）/（BP）/（SI）/（DI）

③ **寄存器相对寻址：**（EA）=（BX）/（BP）/（SI）/（DI）+ DISP8 / DISP16

④ **基址变址寻址：**（EA）=（BX）/（BP）+（SI）/（DI）

⑤ **相对基址变址寻址：**（EA）=（BX）/（BP）+（SI）/（DI）+ DISP8 / DISP16

5、段前缀使用的 3 个例外情况

① 串处理指令目的串必须在 ES

② POP 的源和 PUSH 的目的必须在 SS

③ 指令必须在 CS

6、 默认段选择规则

访存类型	所用段及段寄存器	缺省选择规则
取指	代码段 CS	用于取指
堆栈	堆栈段 SS	所有进栈、出栈，任何使用 BP 作为基址寄存器的访存
局部数据	数据段 DS	除相对于堆栈和串处理指令目的串之外的所有数据访问
目的串	附加段 ES	串处理指令的目的串

7、 双操作数指令的 2 个操作数可以都是存储器寻址吗

不可以

8、 立即寻址和直接寻址都用到数值，汇编指令中如何区分

立即寻址所用数值不需要用[]括起来，直接寻址需要

9、 了解符号地址的使用方式

10、寄存器间接寻址使用的寄存器可以是

只能为 BX、BP、SI 或 DI

11、掌握有效地址由 2 种以上成分时的各种表示方法

① 寄存器相对寻址

（EA）=（BX）/（BP）/（SI）/（DI）+ DISP8 / DISP16

② 基址变址寻址

（EA）=（BX）/（BP）+（SI）/（DI）

③ 相对基址变址寻址

（EA）=（BX）/（BP）+（SI）/（DI）+ DISP8 / DISP16

12、学会使用与转移地址有关的寻址方式

I、段内直接转移

① jmp short opr: (IP) <- (IP) + 8 位偏移地址

② jmp near ptr opr: (IP) <- (IP) + 16 位偏移地址

II、段内间接转移

① jmp word ptr opr: (IP) <- (EA)

② jmp reg: (IP) <- (EA)

III、段间直接转移

① jmp far ptr opr:

(IP) <- opr 的段内偏移地址

(CS) <- opr 所在段的段地址

IV、段剑间接转移

① jmp dword ptr opr:

(IP) <- (EA)

(CS) <- (EA + 2)

13、段内直接寻址方式有哪 2 种？段间直接寻址呢？

同上

14、与地址相关的间接寻址如果使用存储器寻址，为什么要指出是字还是双字

内存操作数类型不明确，需要用操作符来明确，避免二义性

15、MOV 指令应注意的细节

① 双操作数指令除串操作外 src，dst 不能同时为 seg 或 mem

② cs. imm 不能做 dst

③ imm 不能传 seg

④ src 与 dst 类型要匹配

16、理解指令 PUSH 和 POP，注意其格式及使用方法

① 活动段成为栈顶，固定端成为栈底

② 堆栈伸展方向是从高地址向低地址

③ 都是字操作

④ SP 始终指向栈顶，即最后进栈的数据

⑤ push reg/mem/seg （SP）←（SP）- 2

⑥ pop reg/mem/seg （SP）←（SP）+ 2

17、XCHG 指令的 2 个操作数中必须有一个是寄存器，对不对

xchg oprd1, oprd2 oprd1: reg/mem oprd2: reg/mem

对的，双操作数两操作数不能同时为内存

18、学会 IN 和 OUT 指令的 2 种使用形式

① 外设端口为 8 位地址

I、in al，port；in ax，port

II、out al，port；out ax，port

② 外设端口为 16 位地址

I、in al，dx；in ax，dx

II、out al，dx；out ax，dx

19、I/O 端口取值范围是多少

0000h~FFFFH

20、了解 XLAT 指令的功能

查表指令

（AL）← （（BX）+（AL））

AL 为 8 位，表格长度不能草果 256；BX 存放表格首地址

21、LEA BX，LIST 与 MOV BX,OFFSET LIST 完成的功能一样吗

一样，但 offset 只能用于符号地址，lea 可计算任何内存寻址方式

Lea ax，[si+di]

22、LDS(LES)指令的功能

lds/les reg, src

(reg) <- src

(ds/es) <- (src + 2)

段寄存器装入指令，src 要求为存储器寻址方式，数据段或附加段的起始地址会发生改变

23、存取标志寄存器的方法有哪些

① **LAHF** load ah with flags 存标志寄存器指令

(AH) <- (flags 的低字节)

② **SAHF** store ah in flags 取标志寄存器指令

(flags 的低字节) <- (AH)

③ **pushf** 标志寄存器进栈指令

sp = sp - 2

④ **popf** 标志寄存器出栈指令

sp = sp + 2

24、8 到 16 位和 16 到 32 的符号扩展使用哪 2 个命令，如何扩展

① **cbw**

change byte to word

将 al 中数据的 符号位 扩展到 ah 中

al 符号位 为 1 --> ah = 0ffh 全 1

② **cwd**

change word to double word

把 ax 中数据的 符号位 扩展到 dx 中

25、INC 和 DEC 指令是否影响 CF 位

不影响，设置除 CF 之外的符号位

26、把数 a 变成-a。用哪个指令完成最简单

neg reg/mem

negate: 否定

求补

(opr) <- -1 * (opr)

按位求反末位加一

(opr) <- 0ffffh - (opr) + 1

27、加减法指令分哪 2 种？乘除法指令分哪 2 种

带进位加减、不带进位加减；有符号乘除、无符号乘除

28、理解加减法后 CF、OF、ZF、SF 如何变化

29、理解 CMP 指令如何完成数的比较

cmp reg/mem, reg/mem

(dst) - (src)

ZF = 1:

dst = src

ZF = 0 && 无符号数 && CF = 1:

dst < src

ZF = 0 && 有符号数 && OF != SF:

dst < src

OF xor SF = 1

此处可以考虑 OF = 0，即没有溢出的情况下，dst - src 对 SF 的值的影响

OF = 0: 无溢出

SF = 1: dst - src < 0 dst < src

30、能够编程完成 2 个双精度数(32 位)加减运算

31、了解 8086 中乘除法操作数寄存器及大小约定

① mul / imul reg/mem

无符号数乘法指令

(ax) <- (al) * (src)

(dx, ax) <- (ax) * (src)

mul dl　　(ax) ← (al)) * (dl)

mul word ptr [3000h]　　(dx, ax) ← (al) * ([2001h], [2000h])

② div / idiv reg/mem

8bits 字节操作

(al) <- (ax) // (src)　　即 商

(ah) <- (ax) mod (src)　即 余数\

16bits 字操作

(ax) <- (dx, ax) // (src)

(dx) <- (dx, ax) mod (src)

修改: 对所有标志无意义

32、理解 cbw 和 cwd 如何配合 idiv 使用

33、理解什么是压缩 BCD 码和非压缩 BCD 码。能够完成 BCD 码的转换

34、学会使用各种逻辑运算指令，理解给出实例

and / or / xor / test

35、各种移位指令的含义

① shl / shr reg/mem, 1/cl

shift left 逻辑左移

无符号数 最高位 -> CF 最低位补 0

② sal / sar reg/mem, 1/cl

arithmetic: 算术 shift left

有符号数 最高位 -> CF 最低位补 0

③ rol / ror reg/mem, 1/cl

rotate left 循环左移

移出位 -> CF && -> 尾

④ rcl / rcr reg/mem, 1/cl

rotate left with carry 带 CF 循环左移

36、移位指令移位次数大于 1 时，使用哪个寄存器

```
mov cl, 2
shr al, cl
```

37、左移指令 SHL 和 SAL 的执行，右移呢

Shl：无符号数 最高位 -> CF 最低位补 0

Sal：有符号数 最高位 -> CF 最低位补 0

Shr：最低位 -> CF 最高位补 0

Sar：低位 -> CF 最高位补符号位

38、学会利用逻辑指令做乘除法

39、掌握各种串操作指令，理解执行过程，能够指出相关寄存器的变化

① movsb / movsw

cld: DF = 0
((ES) : (DI)) <- ((DS) : (SI))
(SI) <- (SI) + 1 / 2
(DI) <- (DI) + 1 / 2

② stosb / stosw

cld
((ES) : (DI)) <- (AL) / (AX)
(DI) <- (DI) + 1 / 2

③ lodsb / lodsw

cld
(AL) <- ((DS) : (SI))
(SI) <- (SI) + 1 / 2

④ cmpsb / cmpsw

cld
((DS) : (SI)) - ((ES) : (DI))
(SI) <- (SI) + 1 / 2
(DI) <- (DI) + 1 / 2

⑤ scasb / scasw

cld
(AL) / (AX) <- ((ES) : (DI))
(DI) = (DI) + 1 / 2

40、REP、REPZ、REPNZ 有什么不同？都分别和那些串处理指令联用？理解其执行过程

① **REP：**循环（CX）次，与 movs、stos 指令连用

② **REPZ：**相等时重复串操作，repeat when equal 要求（CX）> 0 且（ZF）= 1，与 scas，cmps 连用

③ **REPNZ：**repeat when not equal，与 scas、cmps 连用

41、MOVS 常用于串拷贝，STOS 常用于串初始化，对不对

对

42、能写出串处理指令的完整形式

43、循环执行串操作指令要做哪些准备工作

① 存放在数据段中的源串首地址放入 SI
② 存放在附加段中的目的串首地址放入 DI
③ 若使用重复前缀，数据串长度放入 CX
④ 建立方向标志

44、LODS 一般不和 REP 联用，为什么

LODS：从串中取指令，不影响标志位；从数据段的串中取一个数放入 AL 或 AX，不需要重复操作

45、段间和段内转移指令都影响什么寄存器

① **段内：**IP

② **段间：**IP、CS

46、条件转移指令的转移范围是多少

目标地址应在本条转移指令下一条指令地址的-128~127 个字节的范围之内

47、能够理解和使用各种条件转移指令

48、会使用循环指令

49、CALL 和 JMP 都完成转移，所以是一样的，对吗

不一样，call 指令在实现转移之前会将返回地址存入堆栈，以便 ret 指令返回；jmp 指令直接跳转

50、指令 ret m 的含义是什么

再额外释放 m 个字节的堆栈空间

51、理解子程序调用指令 call 对堆栈的影响

① **call near ptr shup：**CS 不变，IP 存入堆栈，（SP）=（SP）- 2

② **call far ptr shup：**CS、IP 存入堆栈，先存 CS，再存 IP。CS 放高地址，IP 放地址，（SP）= （SP）- 4

52、8086 有多少种中断?有哪 2 种？硬件中断有哪 2 种，软中断指令是怎样的？

8086 可响应 256 种中断，分为硬件中断（外部中断）和软件中断（内部中断）。

硬件中断又分为可屏蔽中断和非可屏蔽中断，为 CPU 之外的外设像 CPU 发出请求

软中断指令为 INT N

53、理解处理中断和子程序调用的不同

相同：

① 都需要保护断点，即下一条指令地址入栈

② 都要跳到子程序或中断服务程序

③ 都可实现嵌套

④ 都要返回主程序

不同：

① 调用子程序发生时间是已知的，中断是未知的

② 子程序为主程序服务，两者是主从关系；中断服务程序与主程序是平行关系

③ 调用子程序不需要专门的硬件电路；中断需要

④ 调用本段的子程序只将 IP 入栈，段间子程序 IP、CS 入栈；中断 CS、IP 一定均入栈

54、中断发生时会清 IF、TF，有什么样的影响

中断发生时，自动将 IF、TF 清零。IF = 0：关闭中断，不允许中断嵌套；TF = 0：不允许单步中断

55、掌握 INT 21H 的几个主要功能

① **01h**：键盘单字符输入，送 al

② **02h**：单字符输出，输出 dl 中的字符

③ **09h**：字符串输出，输出首地址在 DX 中的字符串

④ **0ah**：字符串输入，送入首地址在 DX 中的字符串

56、CF、DF、IF 的标志处理指令都是什么

① CF：

I、CLC（clean carry）：进位位置 0

II、CMC（complement:补 carry）：进位位求反

III、STC（set carry）：进位位置 1

② DF：

I、CLD (clear direction): 方向标志置 0 指令，递增

II、STD (set direction): 方向标志置 1 指令，递减

③ IF

I、CLI (clear interrupt): 中断标志置 0 指令，关中断

II、STI (set interrupt): 中断标志置 1 指令

汇编语言程序格式

指令：能够产生目标代码，是 CPU 可以执行完成特定功能的语句，在汇编时一条指令语句被翻译成特定的机器码来完成相关操作

伪指令：为汇编和连接程序提供一些必要控制的管理型语句，不产生目标代码，仅在汇编过程中高数程序应如何汇编，并完成响应的伪操作

标识符：

① **指令：**标号，后跟":"

② **伪指令：**名字，后跟" "

操作项：指令助记符或伪指令符，表名该语句的操作类型或汇编程序要完成的操作

操作数：操作项的操作对象

1、 伪指令是用来执行的，对吗？它和机器指令有对应关系吗

不是，没有对应关系

2、 掌握汇编程序的基本格式

3、 在代码段的开始，一般要完成段寄存器赋值，这其中是否包括 CS

不包括

4、 assume 语句是否能完成段寄存器 DS、ES 等的赋值

将某一个段寄存器设置为某一个逻辑地址，即明确指出源程序中逻辑段与物理段之间的关系，不能完成段寄存器的赋值

5、 能够读懂简化格式编写的汇编程序

6、 学会如何定义数组

7、 End 伪指令后面一定要跟一个标号吗？说明理由

End [label]

End 为表示源程序结束的伪操作，label 指示程序开始执行的起始地址，若多个程序模块相连接，只有主程序要使用标号，其他的子程序模块只用 end 而不必指定标号

8、 和标号名不同，变量名后面没有冒号，对吗

对的，变量名后面是" "

9、 理解 db、dw、dd 的使用

① db define byte，定义字节，其后的每个操作数都占 1 字节

② dw define word，定义字，其后的每个操作数都占 1 个字

③ dd define double word 定义双字，其后的每个操作数都占 2 个字

10、dup 的使用

dup 复制操作符 repeat-count dup（operand）

重复括号里面的内容 repleat-count 次

11、如何用 dw 和 dd 存符号地址

① **dw：**变量或标号的偏移地址存入存储器

② **dd：**把 16 位段地址、16 位偏移地址存入寄存器。第一个字为偏移地址，位于低地址，第二个字为段地址，位于高地址

12、用 PTR 属性操作符能改变变量隐含的类型属性吗？？？

不能，是一种临时性的属性？？？

13、学会 label 的使用

Label 用来定义属性，只定义类型，不赋处置

① 数据项 variable-name label type（bype / word / dword）

② 可执行代码 label-name label type（near / far）

14、学会 equ 的使用，equ 和=是否等价？

Expression-name equ express

不等价，equ 表达式名不允许重复定义，=允许重复定义

```
Con equ 256
B equ [si + 8]
C equ ds:[bp+9]
D equ con + 2

Emp = 7
Emp = emp + 1
```

15、equ 表达式中使用变量时的对变量有什么要求

表达式中若含有变量或标号，应该在该语句之前给出它们的定义

16、$一定表示地址计数器的当前值吗？为什么？

不一定

① $ 在**指令**中表示本指令第一个字节的地址

Jne $+6；$+6 必须为另一条指令的首地址

② $ 在**伪操作**的参数字段，表示地址计数器的当前的值

13、学会 ORG 的使用

Org constant expression n

设置地址计数器的值，是下一字节的地址为常数表达式的值 n

定义一个 8 字节的数据缓冲区：

```
Buffer label byte
Org $ + 8
```

14、编写汇编程序时十进制数不需后缀，为什么

不需要，汇编程序默认的数为 10 进制数，除非专门指定，汇编程序会把程序中出现的数一律当作 10 进制数

10 进制：D

16 进制：H

2 进制：B

15、了解标号名和变量名字的起名规则

① 合法字符：A~Z，a~z，0~9，？._$

② 除数字外，均可放在首位

③ 名字中用到.，则"."必须在首位

④ 可以用很多字符来声明名字，但汇编程序只识别前 31 个

16、比较一下变量和标号的异同

① 相同

I、起名规则相同

II、均具有段属性（段起始地址）、偏移属性（段内偏移地址）、类型属性三种属性

② 不同

I、标号在代码段中定义；变量在数据段或附加段中定义

II、标号后面跟":"；变量后面跟" "

III、标号的类型属性是 far 或者 near；变量是 byte、word 或者 double word

IV、标号的段属性总是在代码段中

17、能够使用最基本的操作符

18、理解"符号地址　常数表达式"和"符号地址相减"含义

① 符号地址+-常数有意义

② 符号地址减符号地址结果为两变量的偏移地址相差的字节数，只能减

19、关系操作结果的真假在机器内如何表示?

要求两个操作数必须全为立即数，或是同一段内的两个存储器地址

为真表示为全 1，0FFFFh（即-1）；为假表示为 0

Mov ax,（offset Y – offset X）le 128

20、掌握数值回送操作符

① **type expression**

expression

变量: 回送该变量字节数

标号: near ret -1; far ret -2

常数: return 0

② **length variable**

variable 使用 dup() ret repeat-count 否则 return 1

③ **size variable**

return 分配给该变量的字节数

= length variable * type variable

④ **offset variable/label**

return 变量 或 标号 偏移地址

⑤ **seg variable/label**

return 变量 或 标号 段地址值

21、在汇编程序中，注释是必须的吗

不是，用来说明伪操作的功能

22、了解以过程(子程序)的形式定义主程序的方式

```
datas segment
datas ends

stacks segment
    a db 2, 3
stacks ends

codes segment
    assume cs: codes, ds: datas, ss: stacks
    main proc far
        start:
            push ds
            sub ax, ax
            push ax

            ret
    main endp
codes ends
end start
```

23、返回 DOS 的 2 种方式(int 21 的 4c 号功能和利用 ret)

```
Mov ah, 4ch
Int 21h
```

24、了解 masm 和 link、debug 的作用

① masm：对源文件汇编，汇编后产生二进制的目标文件(OBJ 文件)

② link：把 OBJ 文件转换为可执行的 EXE 文件

③ debug：专门为汇编语言设计的一种调试工具，它通过单步执行、设置断点等方式为汇编语言程序员提供了非常有效的调试手段

25、COM 文件所占有的空间不允许超过 64KB，对吗？

对的，.com 文件只能有一个段

编程部分

1、双精度数计算

① 加法和减法

```
datas segment
    x dd 64
    y dd 32
    z dd 12
datas ends

codes segment
    assume cs: codes, ds: datas
    main proc far
        start:
            mov ax, datas
            mov ds, ax

            xor ax, ax
; x，y，z 是双精度数，
    ; 实现加法：x + y + 24 - z
            mov ax, word ptr [x]
            mov dx, word ptr [x + 2]
            add ax, word ptr [y]
            adc bx, word ptr [y + 2]

            add ax, 24
            adc dx, 0

            sub ax, word ptr [z]
            sbb dx, word ptr [z + 2]

            nop

            mov ah, 4ch
            int 21h
        main endp
    codes ends
end start
```

② 16 位乘除法

```
datas segment
    v dw 24
    x dw 16
    y dw 4
    z dw 32
datas ends

codes segment
    assume cs: codes, ds: datas
    main proc far
        start:
            mov ax, datas
            mov ds, ax

            xor ax, ax

            mov ax, [x]
            imul [y]

            mov cx, ax
            mov bx, dx

            mov ax, [z]
            cwd
            add cx, ax
            adc bx, dx

            sub cx, 540
            sbb dx, 0

            mov ax, [v]
            cwd
            sub ax, cx
            sub dx, bx

            idiv [x]

            nop
            mov ah, 4ch
            int 21h
        main endp
    codes ends
end star
```

2、移位运算实现乘除法

3、串处理指令相关程序

1、查找某一个元素是否存在于串中

```
datas segment
    s db '1234acbd'
datas ends

codes segment
    assume cs: codes, ds: datas, es: datas
    main proc far
        start:
            mov ax, datas
            mov ds, ax
            mov es, ax

            xor ax, ax

            mov al, 'a'
            mov cx, 8
            mov bx, cx

            cld
            lea di, s
            repnz scasb
            jz find
            jmp exit

        find:
            sub di, type s
            ; 所在单元的地址: di
            sub bx, cx
            ; 在字符串的位置: bx

        exit:
            mov ah, 4ch
            int 21h

            nop
            mov ah, 4ch
            int 21h
    main endp
codes ends
end start
```

2、找到两个串不匹配的位置，并取出数据

```
datas segment
    s_one db '1234abcde'
    s_two db '1234abdde'
datas ends

codes segment
    assume cs: codes, ds: datas, es: datas
    main proc far
        start:
            mov ax, datas
            mov ds, ax
            mov es, ax

            xor ax, ax

            lea si, s_one
            lea di, s_two
            mov cx, 9
            mov bx, cx

            cld
            repz cmpsb
            jnz not_match
            jmp exit

        not_match:
            sub si, type s_one
            sub di, type s_one
            lodsb
            ; 所在单元的地址: di，si
            sub bx, cx
            ; 在字符串的位置：bx

        exit:
            nop
            mov ah, 4ch
            int 21h

    main endp
codes ends
end start
```

3、将一个串整体后移 shift_number 位

```
    datas segment
        s_one db '1234abcde'
        shift_number dw 9
    datas ends

    codes segment
        assume cs: codes, ds: datas, es: datas
        main proc far
            start:
                mov ax, datas
                mov ds, ax
                mov es, ax

                xor ax, ax

                lea si, s_one
                add si, 8

                mov di, si
                add di, [shift_number]
                mov cx, 9

                std
                rep movsb

            exit:
                nop
                mov ah, 4ch
                int 21h

        main endp
    codes ends
end start
```

3、串的初始化，存入串指令

```
datas segment
    s db 10 dup (0)
datas ends

codes segment
    assume cs: codes, ds: datas, es: datas
    main proc far
        start:
            mov ax, datas
            mov ds, ax
            mov es, ax

            xor ax, ax

            lea di, s
            mov cx, 10
            cld
            mov al, 'a'
            rep stosb

        exit:
            nop
            mov ah, 4ch
            int 21h

    main endp
codes ends
end start
```

4、 ASCII 与数字转换

5、依据某种条件进行统计，如 0 1 个数、符合条件字符个数等等

统计 1 的个数

```
datas segment
datas ends

codes segment
    assume cs: codes, ds: datas, es: datas
    main proc far
        start:
            mov ax, datas
            mov ds, ax
            mov es, ax

            xor ax, ax

            mov al, 10110110b
            mov dx, 0
            mov cx, 8

        calcu:
            shr al, 1
            adc dx, 0
            loop calcu

        show:
            and dx, 00ffh
            add dl, '0'
            mov ah, 02h
            int 21h
        exit:
            nop
            mov ah, 4ch
            int 21h

    main endp
codes ends
end start
```

6、数组元素的查找、插入、删除、排序

冒泡排序：

```
datas segment
    array db 1, 2, 3
datas ends

codes segment
    assume cs: codes, ds: datas, es: datas
    main proc far
        start:
            mov ax, datas
            mov ds, ax
            mov es, ax

            xor ax, ax

            lea bx, array
            mov si, 0
            mov di, 0

        reset:
            cmp si, 2
            je exit
            add si, 1
            mov di, 0
        sort:
            cmp di, si
            je reset

            mov al, [bx + si]
            mov cl, [bx + di]
            cmp al, cl
            jb change
        back:
            add di, 1
            jmp sort

        change:
            xchg al, cl
            mov [bx + si], al
            mov [bx + di], cl
            jmp back

        exit:
            nop
            mov ah, 4ch
            int 21h

    main endp
codes ends
end start
```

7、 键盘输入和显示输出

8、2 个数组之间的运算

9、递归程序的理解，对堆栈的占用