

题型

一、单项选择题（10 分）10道（从下面的知识点里出）

二、分析与简答题（共 8 题，共 43 分）

三、编程与综合题（共 4 题，共 47 分）

1、存储器扩展，根据指定地址范围确定扩展芯片数量，写出每片地址（画地址表），全译码方式下画出总线连接图（题目会给出38译码器（74LS138）、芯片逻辑图）。（15分）作业题5-12

2、8253 方式 0、1、2 和 3、4、5，一个计数通道或两个计数通道级联，初始化编程（要会计算计数初值）。（10分）

3、反转法识别按键编程（12分）（习题 10.12，会改地址范围和接口）

4、编程题。（10分）下面知识点20（不会出原题）

试卷上会给的附录：（不用记忆方式几的控制字怎么写了）

【附 1】：8253 控制字格式：

SC1	SC0	RL1	RL0	M2	M1	M0	BCD
-----	-----	-----	-----	----	----	----	-----

SC1、SC0：选择通道 0(00)、1(01)、2(10)；

RL1、RL0：读/写操作位，定义对选中通道中的计数器的读/写操作方式：

00 计数器锁存，供 CPU 读； 01 读/写计数器低字节；

10 读/写计数器高字节； 11 先读/写计数器低字节，后读/写计数器高字节；

M2、M1、M0：工作方式选择位；

000 计数结束中断方式； 001 可编程单稳态输出方式； 010

频率发生器；

011 方波发生器；

100 软件触发选通；

101 硬件触发选通；

BCD：计数方式选择位。BCD=1，采用 BCD 码计数；BCD=0，采用二进制计数。

【附 2】：8255A 方式选择控制字

1	D6	D5	D4	D3	D2	D1	D0
---	----	----	----	----	----	----	----

D6、D5：A 组方式选择，00 为方式 0，01 为方式 1，1×为方式 2；

D4：A 口 I/O，D4=1 为输入，D4=0 为输出；

D3：C 口高 4 位 I/O，D3=1 为输入，D3=0 为输出；

D2：B 组方式选择，D2=0 为方式 0，D2=1 为方式 1；

D1：B 口 I/O，D1=1 为输入，D1=0 为输出；

D0：C 口低 4 位 I/O，D0=1 为输入，D0=0 为输出。

【附 3】：8255A 端口 C 置位/复位控制字

0	D6	D5	D4	D3	D2	D1	D0
---	----	----	----	----	----	----	----

D6、D5、D4：任意值；D3、D2、D1：位选择，

000 选中 PC0，001 选中 PC1，……，111 选中 PC7；

D0：D0=1 为置位，D0=0 为复位。

【附 4】：8259A 初始化命令字

初始化字 ICW1	0	0	0	1	LTIM	0	SNGL	IC4
中断向量字 ICW2	T7	T6	T5	T4	T3	0	0	0
级联命令字 ICW3	S7	S6	S5	S4	S3	S2/ID2	S1/ID1	S0/ID0
中断方式字 ICW4	0	0	0	SFNM	BUF	M/S	AEOI	μPM
屏蔽命令字 OCW1	M7	M6	M5	M4	M3	M2	M1	M0
中断结束字 OCW2	R	SL	EOI	0	0	L2	L1	L3

注：A0=0 时，写入 ICW1、OCW2 和 OCW3；A0=1 时，写入 ICW2~ICW4 和 OCW1

考察知识点:

1. 存储器操作数寻址方式的分类。

直接寻址

寄存器间接寻址

寄存器相对寻址

基址变址寻址

相对基址变址寻址

2. 常用的寄存器的用途，比如 AX、BX、CX、DX、CS、DS、IP 等。

- **AX (累加器)**: 算术运算、IO 操作的缺省寄存器。
- **BX (基址寄存器)**: 常用作存储器地址的基址。
- **CX (计数寄存器)**: 循环计数、字符串操作计数。
- **DX (数据寄存器)**: 双精度运算、IO 端口地址。

指针和变址寄存器

- **SI (源索引)**: 用于源变址，只能 16 位操作。
- **DI (目的索引)**: 用于目的变址，只能 16 位操作。
- **SI (源变址)**: 子程序中操作的源地址。
- **DI (目的变址)**: 子程序中操作的目的地址。

段寄存器 (16 位):

- **CS (代码段)**: 存放当前程序代码的段地址。
- **DS (数据段)**: 存放数据的段地址。

SS (堆栈段): 存放堆栈的段地址

ES (额外段): 额外的数据段地址。

特殊寄存器:

- **IP (指令指针)**: 指向下一条要执行的指令。

3. 常用的 DOS 功能调用

1. 字符输入 (1号调用)

- 功能：从键盘输入一个字符，并在屏幕上回显。
- 入口参数：AH = 01H
- 出口参数：AL = 输入字符的 ASCII 码
- 代码写法：

```
MOV AH, 01H  
INT 21H
```

2. 字符输出 (2号调用)

- 功能：在屏幕上显示一个指定的字符。
- 入口参数：AH = 02H, DL = 待输出字符的 ASCII 码
- 代码写法：

```
MOV AH, 02H  
MOV DL, 'A'  
INT 21H
```

3. 字符串输出 (9号调用)

- 功能：输出一段以 \$ 结尾的字符串。
- 入口参数：AH = 09H, DS:DX = 字符串的首地址
- 代码写法：

```
MOV AH, 09H  
LEA DX, 字符串名  
INT 21H
```

4. CPU 访问存储器和访问外设的异同。

都靠地址，访问存储器要求精准访问每个存储单元，每个存储单元都有一个确定的地址。访问外设不需要

存储器的分类。

RAM, ROM

I/O 接口和 I/O 端口的区别。

一个接口芯片中有多个不同的端口，每个端口有明确的地址

5. 常用的算术运算指令和逻辑运算指令的功能、特点、用途。

算术指令特点

加法指令：

- ADD dst, src: $\text{dst} = \text{dst} + \text{src}$, 影响所有状态标志
- ADC dst, src: $\text{dst} = \text{dst} + \text{src} + \text{CF}$, 带进位加法
- INC dst: $\text{dst} = \text{dst} + 1$, 不影响 CF 标志

减法指令：

- SUB dst, src: $\text{dst} = \text{dst} - \text{src}$
- SBB dst, src: $\text{dst} = \text{dst} - \text{src} - \text{CF}$, 带借位减法
- DEC dst: $\text{dst} = \text{dst} - 1$, 不影响 CF 标志
- NEG dst: $\text{dst} = 0 - \text{dst}$, 求补码

乘除法指令：

- MUL src: $\text{AX} = \text{AL} \times \text{src}$ (8 位) 或 $\text{DX:AX} = \text{AX} \times \text{src}$ (16 位)
- DIV src: $\text{AL} = \text{AX} \div \text{src}$, $\text{AH} = \text{AX} \bmod \text{src}$ (8 位)

双精度运算：

- 用于处理 32 位数据的加减法
- 先用 ADD 处理低 16 位，再用 ADC 处理高 16 位

逻辑指令

逻辑运算：

- AND dst, src: 按位与，常用于清零特定位
- OR dst, src: 按位或，常用于置位特定位
- XOR dst, src: 按位异或，相同位异或可清零
- NOT dst: 按位取反

移位指令：

- SHL dst, count: 逻辑左移，低位补 0
- SHR dst, count: 逻辑右移，高位补 0
- SAR dst, count: 算术右移，保持符号位
- ROL dst, count: 循环左移
- ROR dst, count: 循环右移

测试指令：

- TEST dst, src: $\text{dst} \& \text{src}$, 只影响标志，不改变操作数
- CMP dst, src: $\text{dst} - \text{src}$, 只影响标志，不改变操作数

6. 子程序的编写。宏的编写。中断服务程序的编写。（3选1出1道题）

子程序

```
BOX proc
    push ax
    cmp al,'A'
    jb next
    cmp al,'Z'
    ja next
    add al,20h
    pop ax
next: ret
BOX endp
```

实现题目要求

宏

```
BOX macro
    local next
    cmp al,'A'
    jb next
    cmp al,'Z'
    ja next
    add al,20h
    next:
endm
```

实现题目要求

中断服务程序

中断名 PROC FAR

PUSH AX

PUSH BX

实现题目要求

MOV AL, 20H

OUT 20H, AL

POP BX

POP AX

IRET

中断名 ENDP

7. 堆栈的有关操作，push 和 pop 指令的执行结果。（出1道题）

PUSH: SP-2, 存数据 (AX, BX不变)

POP: 取数据, SP+2

ss: sp
2000:00FCH ???
2000:00FDH ???
2000:00FEH ???
2000:00FFH ???
2000:0100H ???

PUSH AX

操作: 将AX(5678H)压入堆栈

执行过程:

1. $SP = SP - 2$

- 原SP = 0100H
- 新SP = 0100H - 2 = 00FEH

2. **数据存入堆栈**

- 低字节(AL=78H)存入[SS:SP] = [2000:00FEH] = 78H
- 高字节(AH=56H)存入[SS:SP+1] = [2000:00FFH] = 56H

POP AX

操作: 从堆栈弹出数据到AX

执行过程:

1. **读取数据**

- 低字节 = [SS:SP] = [2000:00FCH] = 00H
- 高字节 = [SS:SP+1] = [2000:00FDH] = 00H
- 弹出值 = 0000H

2. $AX = 0000H$

3. $SP = SP + 2$

- 原SP = 00FCH
- 新SP = 00FCH + 2 = 00FEH

8. 微处理器的定义。

微处理器是一种用于执行通用计算任务的芯片，具有较高的计算能力和灵活性，可以执行各种不同的指令和操作

9. 冯·诺依曼存储程序工作原理（出1道题）

冯·诺依曼体系结构

五大组成部分:

1. 运算器 (ALU): 执行算术和逻辑运算。
2. 控制器: 控制程序的执行流程。
3. 存储器: 存储程序和数据。
4. 输入设备: 向计算机输入数据和程序。
5. 输出设备: 输出计算结果。

核心思想:

- 程序存储: 程序以二进制形式存储在内存中。
- 程序控制: 计算机自动执行程序。
- 指令顺序执行: 除非遇到跳转指令。



10. 微机的总线结构的好处，使用特点。总线定义，分类。

总线系统

定义：连接计算机各部件的公共通信线路。

分类：

数据总线

地址总线

控制总线

优点：

各部件共享总线，减少连线数量。
便于系统扩展和维护。
便于不同厂家设备兼容。

11. 8086 微处理器的内部结构EU、BIU 的定义和作用，流水线。

EU执行，BIU外部设备进行接口、取指令

流水线工作：

- BIU 负责取指令放入队列，EU 从队列取指令执行。
- 取指和执行并行进行，提高处理效率。

12. 8086 总线周期的构成，每一个 T 状态的主要工作。基本的存储器读、写总线周期构成。

8086 总线周期

总线周期组成：

- T1：输出地址，ALE 有效，地址锁存。
- T2：移除地址，准备数据传输，检查 READY 信号。
- T3：数据传输，读操作时 CPU 接收数据。
- T4：结束周期，准备下一个周期。

基本读周期：

1. T1：CPU 输出地址到地址总线，ALE=1 锁存地址。
2. T2：撤除地址， $\overline{\text{MEMR}}$ 或 $\overline{\text{IOR}}$ 有效。
3. T3：从数据总线读取数据。
4. T4：控制信号无效，准备下一周期。

基本写周期：

1. T1：输出地址，ALE 有效。
2. T2：输出数据到数据总线， $\overline{\text{MEMW}}$ 或 $\overline{\text{IOW}}$ 有效。
3. T3：维持数据和控制信号。
4. T4：撤除所有信号。

常用控制信号。

片选信号，最大组态读和写为MEMR、MEMW，最小组态读和写为RD非、WR非

13. 存储器地址的译码问题。

低位直接选中存储单元里面的每一个，高位经过译码选择每一个存储芯片



14. 8088 的寻址方式。

立即寻址、寄存器寻址、基址寻址、变址寻址、基址变址寻址、直接寻址、寄存器间接寻址（与1.一样）

对存储器进行分段管理的原因。（出题）

物理、结构的原因。CPU是16位的，里面的寄存器是16位的，有20根地址总线，要形成20位的物理地址才能访问确切的存储单元。通过分段，把每个存储单元的物理地址分成两部分，段地址用段寄存器存，偏移地址用普通寄存器存，两者用特殊的加法相加。

物理地址和逻辑地址的定义。两者之间转换。（出题）

物理地址：访问存储器的实际地址，一个存储单元对应唯一的物理地址

逻辑地址：由程序产生的与段相关的偏移地址部分

物理地址 = 段地址 \times 16 + 偏移地址

15. 常用指令的判断正误。指令执行结果的判断。（选择题，选出对的或错的）

常见错误类型：

1. 操作数类型不匹配：

- MOV AL, 1234H ✗ (AL 是 8 位, 1234H 是 16 位)
- MOV AL, 12H ✓

2. 寻址方式错误：

- MOV [BX], [SI] ✗ (不能内存到内存)
- MOV AX, [BX]; MOV [SI], AX ✓

3. 段寄存器操作：

- MOV CS, AX ✗ (CS 不能直接赋值)
- MOV DS, AX ✓ **MOV DS, 1000H (✗)**

4. 立即数限制：

- ADD [BX], [SI] ✗
- ADD BX, SI ✓ **ADD [BX], 5(✗)**

PUSH AL (✗) SHR AX, 3 (✗) XCHG CS (✗, 后面是什么S都错)

1. 初始: $AX=1234H$, $BX=5678H$

执行: $XCHG\ AH, BL$

问: $AX=?$, $BX=?$

$AX=7834H$, $BX=5612H$

初始: $AX=00C5H$, $BL=11H$

执行: $DIV\ BL$

问: $AL=?$, $AH=?$

• A) $AL=0BH$, $AH=0AH$

• B) $AL=0CH$, $AH=00H$

• ☒ 答案: A

• 解析: 8位除法: AX / BL 。十六进制→十进制 $00C5H=197$, $11H=17$ 。 $197/17=11$ 余 10 。所以商 $AL=0BH$, 余数 $AH=0AH$ 。

初始: $SI=1000H$, $DI=2000H$, $CX=0005H$, $DF=0$

执行: $REP\ MOVSB$

问: 此指令完成什么功能?

• A) 从 $DS:SI$ 指向的内存复制5个字节到 $ES:DI$

16. 8086 的寄存器结构

8086 寄存器详解

数据寄存器 (16 位, 可拆分为 8 位):

- **AX (累加器)**: 算术运算、IO 操作的缺省寄存器。
- **BX (基址寄存器)**: 常用作存储器地址的基址。
- **CX (计数寄存器)**: 循环计数、字符串操作计数。
- **DX (数据寄存器)**: 双精度运算、IO 端口地址。

指针和变址寄存器:

- **SP (栈指针)**: 指向栈顶, 只能 16 位操作。
- **BP (基指针)**: 访问栈中数据的基址。
- **SI (源变址)**: 字符串操作的源地址。
- **DI (目的变址)**: 字符串操作的目的地址。

段寄存器 (16 位):

- **CS (代码段)**: 存放当前程序代码的段地址。
- **DS (数据段)**: 存放数据的段地址。
- **SS (栈段)**: 存放栈的段地址。
- **ES (附加段)**: 额外的数据段地址。

特殊寄存器:

- **IP (指令指针)**: 指向下一条要执行的指令。
- **FLAGS (标志寄存器)**: 存储程序状态和控制信息。

标志寄存器中每一个标志位的含义及应用。

状态标志 状态标志 (=1的条件)

- CF 进位: • CF=1: 无符号数运算有进位/借位
- ZF 零标志 • ZF=1: 运算结果为0
- SF 符号: • SF=1: 运算结果最高位为1 (有符号数为负)
- OF 溢出: • OF=1: 有符号数运算结果超出范围
- PF 奇偶: • PF=1: 结果低8位中1的个数为偶数
- AF 辅助: • AF=1: 低4位向高4位有进位/借位

控制标志 控制标志 (=1的含义)

- DF 方向: • DF=1: 串操作地址递减
- IF 中断: • IF=1: 允许屏蔽中断
- TF 陷阱: • TF=1: 单步调试模式

8086 复位时各寄存器的初始状态。

寄存器	复位值
CS	FFFFH
IP	0000H
DS, SS, ES	0000H
FLAGS	0000H
第一条指令地址	FFFF0H
通用寄存器	随机



17. 微处理器的指令系统。要熟悉常用的伪指令 (比如如何分配存储单元) 和硬指令的功能。形式类似习题 3.9、3.10、3.14、3.15。

18. 8259A 中 IRR、IMR 和 ISR 三个寄存器的含义

IRR (中断请求寄存器):

- 存储当前的中断请求状态
- 每位对应一个中断输入

ISR (服务中断寄存器):

- 存储正在服务的中断
- 用于优先权管理

IMR (中断屏蔽寄存器):

- 控制各中断的屏蔽状态
- 1=屏蔽, 0=允许

19. 中断的概念。

CPU暂停当前程序，转去处理紧急事件，处理完后返回继续执行

中断向量表的定义。

中断向量表是计算机系统中存放各类中断服务程序入口地址的固定连续内存表。

掌握中断向量表的构成，计算中断类型号，中断服务程序入口地址。

位置与大小:

- 物理地址: 00000H ~ 003FFH (前 1KB)
- 共 256 个中断向量, 每个 4 字节

向量结构 (4 字节):

- 字节 0-1: IP (偏移地址)
- 字节 2-3: CS (段地址)

向量地址 = 中断号 \times 4

入口地址 = CS:IP

已知 8086 系统中某个中断服务程序的入口地址存放在内存 0060H~0063H 处，该区域内容如下：

内存地址	内容
0060H	20H
0061H	10H
0062H	00H
0063H	40H

请计算：

1. 这个中断的**中断类型号**是多少？
2. 中断服务程序的**入口地址**（CS:IP）是什么？

1. 计算中断类型号

已知中断向量在内存地址 0060H 处，根据公式：

$$\text{中断向量地址} = \text{中断类型号} \times 4$$

所以：

$$\text{中断类型号} = \frac{0060H}{4} = \frac{96}{4} = 24 = 18H$$

中断类型号 = 18H

2. 提取入口地址

中断向量的 4 字节结构：**IP 在前，CS 在后**

从内存读取：

- 0060H: 20H (IP 低字节)
- 0061H: 10H (IP 高字节) → **IP = 1020H**
- 0062H: 00H (CS 低字节)
- 0063H: 40H (CS 高字节) → **CS = 4000H**

入口地址 = 4000:1020H

中断的工作过程。可屏蔽中断和不可屏蔽中断的响应条件。

中断工作过程：

请求，设备发中断信号

响应，CPU 检测并响应

保护，自动压栈 FLAGS、CS、IP

执行，转中断服务程序

返回，IRET 恢复现场

响应条件：

可屏蔽中断：IF=1

不可屏蔽中断：NMI引脚有信号

8259 的初始化编程（分析与简答出题，一个芯片，不级联）

```
MOV AL, ..B
```

```
OUT 20H, AL
```

```
MOV AL, ..B
```

```
OUT 21H, AL
```

```
MOV AL, ..B
```

```
OUT 21H, AL
```

红色部分固定，只根据题目改黑色部分

第一个：查表ICW1，LTIM边沿出发0（默认），电平出发1。最后两个为1

第二个：查表ICW2，题目会给“中断号从...开始”，转成二进制，高五位对应T7-T3

第三个：查表ICW4，AEOI正常EOI填0（默认），自动EOI填1。前全为0，后为1

20. 循环结构、顺序结构和分支结构编写小程序。（6选2-3个出题，最后一道大题出）（不会出原题）

大小写转换：

小写转大写（大写转小写：a、z换成A、Z, SUB换成ADD）

```
CMP AL, 'a'
JB BOX
CMP AL, 'z'
JA BOX
SUB AL, 20H
BOX:
```

Copy

从一组数中找到最大和最小的数；

（此处是找最大值, 找最小是把JGE换成JLE）

```
MOV AX, [BX]
MOV CX, N-1
LOOP1:
ADD BX, 2
CMP AX, [BX]
JGE CONTINUE
MOV AX, [BX]
CONTINUE:
LOOP LOOP1
```

实现循环累加：（1加到100）

```
mov ax,0
```

```
mov cx,1
```

```
loop1:
```

```
add ax,cx
```

```
add cx,1
```

```
cmp cx,100
```

```
jle loop1
```

求一个数的绝对值；（假设这个数为2）

```
MOV AX, 2
CMP AX, 0
JGE BOX
NEG AX
BOX :
```

Copy

查表法求一个十六进制数的ASCII码；

```
BOX DB '0123456789ABCDEF'

MOV BX, OFFSET BOX
MOV AL, AH
XLAT
```

要转换的十六进制数

Copy

统计一组数中满足一定条件的数的个数
（统计正、负、零的个数）

```
mov si,offset array
```

```
mov cx,count
```

```
mov ax,0
```

```
mov bx,0
```

```
mov dx,0
```

```
loop1:
```

```
mov bp,[si]
```

```
cmp bp,0
```

```
jg positive
```

```
jl negative
```

```
add dx,1
```

```
jmp continue
```

```
positive:
```

```
add ax,1
```

```
jmp continue
```

negative:

add bx,1

jmp continue

continue:

add si,2

loop loop1



22. 计算机主机和 I/O 设备之间进行数据传送的方法。

直接传送，查询传送，中断传送

查询方式完成数据传送的流程，要会编程。

状态口，数据口，看状态口里的外设是否准备好，对外设进行输入输出操作，传送两个方向，读和写。

输入（读）

WAIT:

IN AL, 状态端口

TEST AL, 80H

JZ WAIT

IN AL, 数据端口

输出（写）

WAIT:

IN AL, 状态端口

TEST AL, 80H

JNZ WAIT

OUT 数据端口, AL

类似处理器总线，存储器芯片也分成数据、地址和控制3类引脚。以存储结构为32K×8的SRAM 62256为例，该芯片应有8个数据引脚、15个地址引脚，3个典型的控制引脚分别是片选、输出允许和写入允许。

$$32K = 32 \times 2^{10} = 2^{15}$$

需要15个地址引脚

($2^n \geq 2^{15}$ ，地址引脚数为n)

〔习题5.13〕

给出图5-28中4个存储器芯片各自占用的地址范围。
如果采用部分译码，要指出重复的地址范围。

4个存储芯片各自的可用地址范围：

芯片号	A19A18	A17A16	A15~A0	地址范围
1	00	全0~全1	全0~全1	00000H~3FFFFH
2	01	全0~全1	全0~全1	40000H~7FFFFH
3	11	×0	全0~全1	C0000H~CFFFFH，重复地址：E0000H~EFFFFH
4	11	×1	全0~全1	D0000H~DFFFFH，重复地址：F0000H~FFFFFH

问题：

在8088的工作过程中，什么情况下会产生 T_w ？
发生在什么具体时刻？

解答：

当8088进行读写存储器或I/O接口时，如果存储器或I/O接口无法满足CPU的读写时序（来不及提供或读取数据时），需要CPU插入等待状态 T_w 。

在读写总线周期的 T_3 和 T_4 之间插入 T_w 。

〔习题3.9〕给出下列语句中，指令立即数（数值表达式）的值：

(1) `mov al,23h AND 45h OR 67h`
67h

(2) `mov ax,1234h/16+10h`
133h

(3) `mov ax,23h SHL 4`
0230h

(4) `mov al,'a' AND (NOT('a'-'A'))`
41h

(5) `mov ax,(76543 LT 32768) XOR 7654h`
7654h

Less Than

〔习题2.13〕

设有4个16位带符号数，分别装在X、Y、Z、V存储单元中，阅读如下程序段，得出它的运算公式，并说明运算结果存于何处。

```
mov ax,X
imul Y
mov cx,ax
mov bx,dx    ;X*Y → bx,cx
mov ax,Z
cwd          ;Z → dx,ax

add cx,ax
adc bx,dx    ;X*Y+Z → bx,cx

sub cx,540
sbb bx,0     ;X*Y+Z-540 → bx,cx
mov ax,V
cwd          ;V → dx,ax

sub ax,cx
sbb dx,bx    ;V-(X*Y+Z-540) → dx,ax
idiv X       ;[V-(X*Y+Z-540)]/X → dx (余数),ax (商)
```



问题:

- 请分别用一条汇编语言指令完成如下功能:

解答:

- (1) 把**BX**寄存器和**DX**寄存器的内容相加, 结果存入**DX**寄存器

add dx,bx

- (2) 用寄存器**BX**和**SI**的基址变址寻址方式把存储器的一个字节与**AL**寄存器的内容相加, 并把结果送到**AL**中

add al,[bx+si]

- (3) 用**BX**和位移量**0B2H**的寄存器相对寻址方式把存储器中的一个字和**CX**寄存器的内容相加, 并把结果送回存储器中

add [bx+0b2h],cx

- (4) 用位移量为**0520H**的直接寻址方式把存储器中的一个字与数**3412H**相加, 并把结果送回该存储单元中

add word ptr [0520h],3412h

- (5) 把数**0A0H**与**AL**寄存器的内容相加, 并把结果送回**AL**中

add al,0a0h



问题:

- 给出下列各条指令执行后的结果, 以及状态标志**CF**、**OF**、**SF**、**ZF**、**PF**的状态。

解答:

		CF	ZF	SF	OF	PF
mov ax,1470h	; AX=1470H	0	0	0	0	0
and ax,ax	; AX=1470H	0	0	0	0	0
or ax,ax	; AX=1470H	0	0	0	0	0
xor ax,ax	; AX=0000H	0	1	0	0	1
not ax	; AX=FFFFH	0	1	0	0	1
test ax,0f0f1h	; AX=FFFFH	0	0	1	0	0