


# 《微机原理与接口技术》复习参考资料

教师：万显荣

复习资料说明：

- 1、标有红色星号“”的内容为重点内容
- 3、本资料末尾附有“《微机原理与接口技术》综合练习题与答案错误修正”和“《微机原理与接口技术》综合练习题与答案中不作要求的部分”，请注意查看。

## 第一章 概述

### 一、计算机中的数制

#### 1、无符号数的表示方法：

##### (1) 十进制计数的表示法

特点：以十为底，逢十进一；  
共有 0-9 十个数字符号。

##### (2) 二进制计数表示方法：

特点：以 2 为底，逢 2 进位；  
只有 0 和 1 两个符号。

##### (3) 十六进制数的表示法：

特点：以 16 为底，逢 16 进位；  
有 0-9 及 A—F（表示 10~15）共 16 个数字符号。

#### 2、各种数制之间的转换

##### (1) 非十进制数到十进制数的转换

按相应进位计数制的权表达式展开，再按十进制求和。（见书本 1.2.3，1.2.4）

##### (2) 十进制数制转换为二进制数制

###### ●十进制 → 二进制的转换：

整数部分：除 2 取余；  
小数部分：乘 2 取整。

###### ●十进制 → 十六进制的转换：

整数部分：除 16 取余；  
小数部分：乘 16 取整。

以小数点为起点求得整数和小数的各个位。

##### (3) 二进制与十六进制数之间的转换

用 4 位二进制数表示 1 位十六进制数

#### 3、无符号数二进制的运算（见教材 P5）

#### 4、二进制数的逻辑运算

特点：按位运算，无进借位

##### (1) 与运算

只有 A、B 变量皆为 1 时，与运算的结果就是 1

##### (2) 或运算

A、B 变量中，只要有一个为 1，或运算的结果就是 1

##### (3) 非运算

#### (4) 异或运算

A、B 两个变量只要不同，异或运算的结果就是 1

## 二、计算机中的码制（重点🔔）

1、对于符号数，机器数常用的表示方法有原码、反码和补码三种。数 X 的原码记作  $[X]_{\text{原}}$ ，

反码记作  $[X]_{\text{反}}$ ，补码记作  $[X]_{\text{补}}$ 。

注意：对正数，三种表示法均相同。  
它们的差别在于对负数的表示。

#### (1) 原码

定义：

符号位：0 表示正，1 表示负；

数值位：真值的绝对值。

注意：数 0 的原码不唯一

#### (2) 反码

定义：

若  $X > 0$ ，则  $[X]_{\text{反}} = [X]_{\text{原}}$

若  $X < 0$ ，则  $[X]_{\text{反}} =$  对应原码的符号位不变，数值部分按位求反

注意：数 0 的反码也不唯一

#### (3) 补码

定义：

若  $X > 0$ ，则  $[X]_{\text{补}} = [X]_{\text{反}} = [X]_{\text{原}}$

若  $X < 0$ ，则  $[X]_{\text{补}} = [X]_{\text{反}} + 1$

注意：机器字长为 8 时，数 0 的补码唯一，同为 00000000

2、8 位二进制的表示范围：

原码：-127~+127

反码：-127~+127

补码：-128~+127

3、特殊数 10000000

●该数在原码中定义为：-0

●在反码中定义为：-127

●在补码中定义为：-128

●对无符号数： $(10000000)_2 = 128$

## 三、信息的编码

1、十进制数的二进制数编码

用 4 位二进制数表示一位十进制数。有两种表示法：压缩 BCD 码和非压缩 BCD 码。

(1) 压缩 BCD 码的每一位用 4 位二进制表示，0000~1001 表示 0~9，一个字节表示两位十进制数。

(2) 非压缩 BCD 码用一个字节表示一位十进制数，高 4 位总是 0000，低 4 位的 0000~1001 表示 0~9

## 2、字符的编码

计算机采用 7 位二进制代码对字符进行编码

(1) 数字 0~9 的编码是 0110000~0111001，它们的高 3 位均是 011，后 4 位正好与其对应的二进制代码（BCD 码）相符。

(2) 英文字母 A~Z 的 ASCII 码从 1000001（41H）开始顺序递增，字母 a~z 的 ASCII 码从 1100001（61H）开始顺序递增，这样的排列对信息检索十分有利。

# 第二章 微机组成原理

## 第一节、微机的结构

### 1、计算机的经典结构——冯·诺依曼结构

(1) 计算机由运算器、控制器、输入设备和输出设备五大部分组成（运算器和控制器又称为 CPU）

(2) 数据和程序以二进制代码形式不加区分地存放在存储器总，存放位置由地址指定，数制为二进制。

(3) 控制器是根据存放在存储器中的指令序列来操作的，并由一个程序计数器控制指令的执行。

### 3、系统总线的分类

(1) 数据总线（Data Bus），它决定了处理器的字长。

(2) 地址总线（Address Bus），它决定系统所能直接访问的存储器空间的容量。

(3) 控制总线（Control Bus）

## 第二节、8086 微处理器

1、8086 是一种单片微处理芯片，其内部数据总线的宽度是 16 位，外部数据总线宽度也是 16 位，片内包含有控制计算机所有功能的各种电路。

8086 地址总线的宽度为 20 位，有 1MB（ $2^{20}$ ）寻址空间。

2、8086CPU 由总线接口部件 BIU 和执行部件 EU 组成。BIU 和 EU 的操作是异步的，为 8086 取指令和执行指令的并行操作体统硬件支持。

3、8086 处理器的启动

### 4、寄存器结构（重点🔔）

8086 微处理器包含有 13 个 16 位的寄存器和 9 位标志位。

4 个通用寄存器（AX, BX, CX, DX）

4 个段寄存器（CS, DS, SS, ES）

4 个指针和变址寄存器（SP, BP, SI, DI）

指令指针（IP）

1)、通用寄存器

(1) 8086 含 4 个 16 位数据寄存器，它们又可分为 8 个 8 位寄存器，即：

- AX → AH, AL
- BX → BH, BL
- CX → CH, CL
- DX → DH, DL

常用来存放参与运算的操作数或运算结果

(2) 数据寄存器特有的习惯用法

- AX: 累加器。多用于存放中间运算结果。所有 I/O 指令必须都通过 AX 与接口传送信息；
- BX: 基址寄存器。在间接寻址中用于存放基地址；
- CX: 计数寄存器。用于在循环或串操作指令中存放循环次数或重复次数；
- DX: 数据寄存器。在 32 位乘法运算时，存放高 16 位数；在间接寻址的 I/O 指令中存放 I/O 端口地址。

2)、指针和变址寄存器

- SP: 堆栈指针寄存器，其内容为栈顶的偏移地址；
- BP: 基址指针寄存器，常用于在访问内存时存放内存单元的偏移地址。
- SI: 源变址寄存器
- DI: 目标变址寄存器

变址寄存器常用于指令的间接寻址或变址寻址。

3)、段寄存器

CS: 代码段寄存器，代码段用于存放指令代码

DS: 数据段寄存器

ES: 附加段寄存器，数据段和附加段用来存放操作数

SS: 堆栈段寄存器，堆栈段用于存放返回地址，保存寄存器内容，传递参数

4)、指令指针 (IP)

16 位指令指针寄存器，其内容为下一条要执行的指令的偏移地址。

5)、标志寄存器

(1) 状态标志:

- 进位标志位 (CF): 运算结果的最高位有进位或有借位，则 CF=1
- 辅助进位标志位 (AF): 运算结果的低四位有进位或借位，则 AF=1
- 溢出标志位 (OF): 运算结果有溢出，则 OF=1
- 零标志位 (ZF): 反映指令的执行是否产生一个为零的结果
- 符号标志位 (SF): 指出该指令的执行是否产生一个负的结果
- 奇偶标志位 (PF): 表示指令运算结果的低 8 位“1”个数是否为偶数

(2) 控制标志位

- 中断允许标志位 (IF): 表示 CPU 是否能够响应外部可屏蔽中断请求
- 跟踪标志 (TF): CPU 单步执行

5、8086 的引脚及其功能 (重点掌握以下引脚)

- AD<sub>15</sub>~AD<sub>0</sub>: 双向三态的地址总线，输入/输出信号
- INTR: 可屏蔽中断请求输入信号，高电平有效。可通过设置 IF 的值来控制。
- NMI: 非屏蔽中断输入信号。不能用软件进行屏蔽。
- RESET: 复位输入信号，高电平有效。复位的初始状态见 P21
- MN/MX: 最小最大模式输入控制信号。

## 第三章 8086 指令系统

说明：8086 指令系统这章为重点章节，对下面列出的指令都要求掌握。

### 第一节 8086 寻址方式

#### 一、数据寻址方式（重点🔒）

##### 1、立即寻址

操作数(为一常数)直接由指令给出

(此操作数称为立即数)

立即寻址只能用于源操作数

例：

```
MOV    AX,    1C8FH
```

```
MOV    BYTE PTR[2A00H], 8FH
```

错误例：

```
× MOV 2A00H, AX ; 错误!
```

指令操作例：MOV AX, 3102H; AX→3102H

执行后，(AH) = 31H, (AL) = 02H

##### 2、寄存器寻址

(1) 操作数放在某个寄存器中

(2) 源操作数与目的操作数字长要相同

(3) 寄存器寻址与段地址无关

例：

```
MOV    AX,    BX
```

```
MOV    [3F00H], AX
```

```
MOV    CL,    AL
```

错误例：

```
× MOV AX, BL ; 字长不同
```

```
× MOV ES:AX, DX ; 寄存器与段无关
```

##### 3、直接寻址

(1) 指令中直接给出操作数的 16 位偏移地址 偏移地址也称为有效地址 (EA, Effective Address)

(2) 默认的段寄存器为 DS，但也可以显式地指定其他段寄存器——称为段超越前缀

(3) 偏移地址也可用符号地址来表示，如 ADDR、VAR

例：

```
MOV    AX, [2A00H]
```

```
MOV    DX, ES:[2A00H]
```

```
MOV    SI, TABLE_PTR
```

##### 4、间接寻址

● 操作数的偏移地址(有效地址 EA)放在寄存器中

- 只有 SI、DI、BX 和 BP 可作间址寄存器

- 例： `MOV AX,[BX]`  
`MOV CL,CS:[DI]`

错误例：  $\times$  `MOV AX,[DX]`  
 $\times$  `MOV CL,[AX]`

## 5、寄存器相对寻址

- EA=间址寄存器的内容加上一个 8/16 位的位移量

- 例： `MOV AX, [BX+8]`  
`MOV CX, TABLE[SI]`  
`MOV AX, [BP]`; 默认段寄存器为 SS

- 指令操作例： `MOV AX, DATA[BX]`

若 (DS)=6000H, (BX)=1000H, DATA=2A00H,  
(63A00H)=66H, (63A01H)=55H

则物理地址 = 60000H + 1000H + 2A00H = 63A00H

指令执行后：(AX) = 5566H

## 6、基址变址寻址

- 若操作数的偏移地址：

由基址寄存器(BX 或 BP)给出 — 基址寻址方式

由变址寄存器(SI 或 DI)给出 — 变址寻址方式

由一个基址寄存器的内容和一个变址寄存器的内容相加而形成操作数的偏移地址，称为基址-变址寻址。

$EA = (BX) + (SI) \text{ 或 } (DI);$

$EA = (BP) + (SI) \text{ 或 } (DI)$

同一组内的寄存器不能同时出现。

注意：除了有段跨越前缀的情况外，当基址寄存器为 BX 时，操作数应该存放在数据段 DS 中，当基址寄存器为 BP 时，操作数应放在堆栈段 SS 中。例：

`MOV AX, [BX][SI]`  
`MOV AX, [BX+SI]`  
`MOV AX, DS:[BP][DI]`

错误例：

$\times$  `MOV AX, [BX][BP]`  
 $\times$  `MOV AX, [DI][SI]`

指令操作例： `MOV AX, [BX][SI]`

假定：(DS)=8000H, (BX)=2000H, SI=1000H

则物理地址 = 80000H + 2000H + 1000H = 83000H

指令执行后：(AL)=[83000H]

(AH)=[83001H]

## 7、相对基址变址寻址

- 在基址-变址寻址的基础上再加上一个相对位移量

EA= (BX) + (SI) 或 (DI) +8 位或 16 位位移量;

EA= (BP) + (SI) 或 (DI) +8 位或 16 位位移量

指令操作例: MOV AX, DATA[DI][BX]

若(DS)=8000H, (BX)=2000H, (DI)=1000H, DATA=200H

则指令执行后(AH)=[83021H], (AL)=[83020H]

寄存器间接、寄存器相对、基址变址、相对基址变址四种寻址方式的比较:

寻址方式	指令操作数形式
■ 寄存器间接	只有一个寄存器 (BX/BP/SI/DI 之一)
■ 寄存器相对	一个寄存器加上位移量
■ 基址—变址	两个不同类别的寄存器
■ 相对基址-变址	两个不同类别的寄存器加上位移量

## 二、地址寻址方式 (了解有 4 类, 能判断)

简要判断依据 (指令中间的单词):

段内直接 short, near

段内间接 word

段间直接 far

段间间接 dword

## 第二节 8086 指令系统

### 一、数据传送指令 (重点🔔)

#### 1、通用传送指令

(1) MOV dest, src; dest ← src

传送的是字节还是字取决于指令中涉及的寄存器是 8 位还是 16 位。

具体来说可实现:

① MOV mem/reg1, mem/reg2

指令中两操作数中至少有一个为寄存器

② MOV reg, data ;立即数送寄存器

③ MOV mem, data ;立即数送存储单元

④ MOV acc, mem ;存储单元送累加器

⑤ MOV mem, acc ;累加器送存储单元

⑥ MOV segreg, mem/reg ;存储单元/寄存器送段寄存器

⑦ MOV mem/reg, segreg ;段寄存器送存储单元/寄存器

#### MOV 指令的使用规则

① IP 不能作目的寄存器

② 不允许 mem ← mem

③ 不允许 segreg ← segreg

④ 立即数不允许作为目的操作数

⑤ 不允许 segreg ← 立即数

⑥ 源操作数与目的操作数类型要一致

⑦当源操作数为单字节的立即数，而目的操作数为间址、变址、基址+变址的内存数时，必须用 PTR 说明数据类型。如：MOV [BX], 12H 是错误的。

## (2)、堆栈指令

什么是堆栈？

按“后进先出(LIFO)”方式工作的存储区域。堆栈以字为单位进行压入弹出操作。

规定由 SS 指示堆栈段的段基址，堆栈指针 SP 始终指向堆栈的顶部，SP 的初值规定了所用堆栈区的大小。堆栈的最高地址叫栈底。

### ① 压栈指令 PUSH

PUSH src ; src 为 16 位操作数

例：PUSH AX ; 将 AX 内容压栈

执行操作：(SP) -1 ← 高字节 AH

(SP) -2 ← 低字节 AL

(SP) ← (SP) - 2

注意进栈方向是高地址向低地址发展。

### ② 弹出指令 POP

POP dest

例：POP BX ; 将栈顶内容弹至 BX

执行操作：(BL) ← (SP)

(BH) ← (SP) + 1

(SP) ← (SP) + 2

堆栈指令在使用时需注意的几点：

- ① 堆栈操作总是按字进行
- ② 不能从栈顶弹出一个字给 CS
- ③ 堆栈指针为 SS:SP，SP 永远指向栈顶
- ④ SP 自动进行增减量 (-2, +2)

## (3)、交换指令 XCHG

格式：XCHG reg, mem/reg

功能：交换两操作数的内容。

要求：两操作数中必须有一个在寄存器中；

操作数不能为段寄存器和立即数；

源和目的地操作数类型要一致。

举例：XCHG AX, BX

XCHG [2000], CL

## (4) 查表指令 XLAT

执行的操作：AL ← [(BX)+(AL)]



---

又叫查表转换指令，它可根据表项序号查出表中对应代码的内容。执行时先将表的首地址（偏移地址）送到 BX 中，表项序号存于 AL 中。

## 2、输入输出指令

只限于用累加器 AL 或 AX 来传送信息。

功能: (累加器) $\longleftrightarrow$ I/O 端口

### (1) 输入指令 IN

格式:

IN acc,PORT ;PORT 端口号 0~255H

IN acc,DX ;DX 表示的端口范围达 64K

例:IN AL, 80H ;(AL) $\leftarrow$ (80H 端口)

IN AL, DX ;(AL) $\leftarrow$ ((DX))

### (2) 输出指令 OUT

格式: OUT port,acc

OUT DX,acc

例: OUT 68H, AX ;(69H, 68H) $\leftarrow$ (AX)

OUT DX, AL ;(DX) $\leftarrow$ (AL)

在使用间接寻址的 IN/OUT 指令时，要事先用传送指令把 I/O 端口号设置到 DX 寄存器如：

MOV DX, 220H

IN AL, DX;将 220H 端口内容读入 AL

## 3、目标地址传送指令

### (1) LEA

传送偏移地址

格式: LEA reg, mem ;将指定内存单元的偏移地址送到指定寄存器

要求:

1) 源操作数必须是一个存储器操作数;

2) 目的操作数必须是一个 16 位的通用寄存器。

例: LEA BX, [SI+10H]

设: (SI) =1000H

则执行该指令后, (BX) =1010H

### ●注意以下二条指令差别:

LEA BX, BUFFER

MOV BX, BUFFER

前者表示将符号地址为 BUFFER 的存储单元的偏移地址取到 BX 中;后者表示将 BUFFER 存储单元中的内容取到 BX 中。

下面两条指令等效:

LEA BX, BUFFER

---

MOV BX, OFFSET BUFFER

其中 **OFFSET BUFFER** 表示存储器单元 BUFFER 的偏移地址。

二者都可用于取存储器单元的偏移地址，但 LEA 指令可以取动态的地址，OFFSET 只能取静态的地址。

## 二、算术运算指令

### 1、加法指令

(1) 不带进位的加法指令 ADD

格式:    ADD       acc,data  
          ADD       mem/reg,data  
          ADD  mem/reg1,mem/reg2

实例:

```
ADD     AL, 30H
ADD     SI, [BX+20H]
ADD     CX, SI
ADD     [DI], 200H
```

•ADD 指令对 6 个状态标志均产生影响。

例: 已知(BX)=D75FH

指令 ADD BX,8046H 执行后, 状态标志各是多少?

D75FH = 1110 0111 0101 1111	
8046H = 1000 0000 0100 0110	
1	1 11 11
<hr/>	
0110 0111 1010 0101	

结果: C=1, Z=0, P=0, A=1, O=1, S=0

### 判断溢出与进位 (重点🔴)

从硬件的角度: 默认参与运算的操作数都是有符号数, 当两数的符号位相同, 而和的结果相异时有溢出, 则 OF=1, 否则 OF=0

(2) 带进位的加法 ADC

ADC 指令在形式上和功能上与 ADD 类似, 只是相加时还要包括进位标志 CF 的内容, 例如:

```
ADC  AL, 68H;  AL←(AL)+68H+(CF)
ADC  AX, CX  ;AX←(AX)+(CX)+(CF)
ADC  BX, [DI] ;BX←(BX)+[DI+1][DI]+(CF)
```

(3) 加 1 指令 INC

格式: INC reg/mem

功能: 类似于 C 语言中的++操作: 对指定的操作数加 1

例:    INC  AL  
          INC  SI  
          INC  BYTE PTR[BX+4]

**注：本指令不影响CF标志。**

(4) 非压缩BCD码加法调整指令AAA

AAA指令的操作：

如果AL的低4位>9或AF=1，则：

- ①  $AL \leftarrow (AL) + 6, (AH) \leftarrow (AH) + 1, AF \leftarrow 1$
- ② AL高4位清零
- ③  $CF \leftarrow AF$

否则AL高4位清零

(5) 压缩BCD码加法调整指令DAA

●两个压缩BCD码相加结果在AL中，通过DAA调整得到一个正确的压缩BCD码。

●指令操作(调整方法)：

若AL的低4位>9或AF=1

则 $(AL) \leftarrow (AL) + 6, AF \leftarrow 1$

若AL的高4位>9或CF=1

则 $(AL) \leftarrow (AL) + 60H, CF \leftarrow 1$

●除OF外，DAA指令影响所有其它标志。

●DAA指令应紧跟在ADD或ADC指令之后。

## 2、减法指令

(1) 不考虑借位的减法指令SUB

格式：SUB dest, src

操作： $dest \leftarrow (dest) - (src)$

注：1. 源和目的操作数不能同时为存储器操作数

2. 立即数不能作为目的操作数

指令例子：

SUB AL, 60H

SUB [BX+20H], DX

SUB AX, CX

(2) 考虑借位的减法指令SBB

SBB指令主要用于多字节的减法。

格式：SBB dest, src

操作： $dest \leftarrow (dest) - (src) - (CF)$

指令例子：

SBB AX, CX

SBB WORD PTR[SI], 2080H

SBB [SI], DX

(3) 减1指令DEC

作用类似于C语言中的“--”操作符。

格式：DEC opr

操作： $opr \leftarrow (opr) - 1$

指令例子：

---

DEC CL  
DEC BYTE PTR[DI+2]  
DEC SI

(4) 求补指令 NEG

格式: NEG opr

操作:  $\text{opr} \leftarrow 0 - (\text{opr})$

对一个操作数取补码相当于用 0 减去此操作数, 故利用 NEG 指令可得到负数的绝对值。

例: 若(AL)=0FCH, 则执行 NEG AL 后,

(AL)=04H, CF=1

(5) 比较指令 CMP

格式: CMP dest, src

操作:  $(\text{dest}) - (\text{src})$

CMP 也是执行两个操作数相减, 但结果不送目标操作数, 其结果只反映在标志位上。

指令例子:

CMP AL, 0AH

CMP CX, SI

CMP DI, [BX+03]

(6) 非压缩 BCD 码减法调整指令 AAS

对 AL 中由两个非压缩的 BCD 码相减的结果进行调整。调整操作为:

若 AL 的低 4 位 > 9 或 AF=1, 则:

①  $\text{AL} \leftarrow (\text{AL}) - 6, \text{AH} \leftarrow (\text{AH}) - 1, \text{AF} \leftarrow 1$

② AL 的高 4 位清零

③  $\text{CF} \leftarrow \text{AF}$

否则: AL 的高 4 位清零

(7) 压缩 BCD 码减法调整指令 DAS

对 AL 中由两个压缩 BCD 码相减的结果进行调整。调整操作为:

若 AL 的低 4 位 > 9 或 AF=1, 则:

$\text{AL} \leftarrow (\text{AL}) - 6$ , 且  $\text{AF} \leftarrow 1$

若 AL 的高 4 位 > 9 或 CF=1, 则:

$\text{AL} \leftarrow (\text{AL}) - 60\text{H}$ , 且  $\text{CF} \leftarrow 1$

DAS 对 OF 无定义, 但影响其余标志位。

DAS 指令要求跟在减法指令之后。

### 3、乘法指令

进行乘法时: 8 位\*8 位 → 16 位乘积

16 位\*16 位 → 32 位乘积

(1) 无符号数的乘法指令 MUL (MEM/REG)

格式: MUL src

操作: 字节操作数  $(\text{AX}) \leftarrow (\text{AL}) \times (\text{src})$

字操作数       $(DX, AX) \leftarrow (AX) \times (src)$

指令例子:

MUL BL ;  $(AL) \times (BL)$ , 乘积在 AX 中  
MUL CX ;  $(AX) \times (CX)$ , 乘积在 DX, AX 中  
MUL BYTE PTR[BX]

(2) 有符号数乘法指令 IMUL

格式与 MUL 指令类似, 只是要求两操作数均为有符号数。

指令例子:

IMUL BL ;  $(AX) \leftarrow (AL) \times (BL)$   
IMUL WORD PTR[SI];  
 $(DX, AX) \leftarrow (AX) \times ([SI+1][SI])$

注意: MUL/IMUL 指令中

- AL (AX) 为隐含的乘数寄存器;
- AX (DX, AX) 为隐含的乘积寄存器;
- SRC 不能为立即数;
- 除 CF 和 OF 外, 对其它标志位无定义。

#### 4、除法指令

进行除法时: 16 位/8 位→8 位商

32 位/16 位→16 位商

对被除数、商及余数存放有如下规定:

	被除数	商	余数
字节除法	AX	AL	AH
字除法	DX:AX	AX	DX

(1) 无符号数除法指令 DIV

格式: DIV src

操作: 字节操作  $(AL) \leftarrow (AX) / (SRC)$  的商  
 $(AH) \leftarrow (AX) / (SRC)$  的余数  
字操作  $(AX) \leftarrow (DX, AX) / (SRC)$  的商  
 $(DX) \leftarrow (DX, AX) / (SRC)$  的余数

指令例子:

DIV CL  
DIV WORD PTR[BX]

(2) 有符号数除法指令 IDIV

格式: IDIV src

操作与 DIV 类似。商及余数均为有符号数, 且余数符号总是与被除数符号相同。

注意: 对于 DIV/IDIV 指令

AX(DX, AX) 为隐含的被除数寄存器。

AL(AX) 为隐含的商寄存器。

---

**AH(DX)为隐含的余数寄存器。**

**src 不能为立即数。**

**对所有条件标志位均无定**

### 关于除法操作中的字长扩展问题

- 除法运算要求被除数字长是除数字长的两倍,若不满足则需**对被除数进行扩展**,否则产生错误。
- 对于无符号数除法扩展,只需将 AH 或 DX 清零即可。
- 对有符号数而言,则是符号位的扩展。可使用前面介绍过的符号扩展指令 CBW 和 CWD

## 三、逻辑运算和移位指令

### 1、逻辑运算指令

#### (1) 逻辑与 AND

对两个操作数进行按位逻辑“与”操作。

格式: AND dest, src

用途: 保留操作数的某几位, 清零其他位。

例 1: 保留 AL 中低 4 位, 高 4 位清 0。

```
AND AL, 0FH
```

#### (2) 逻辑或 OR

对两个操作数进行按位逻辑“或”操作。

格式: OR dest, src

用途: 对操作数的某几位置 1; 对两操作数进行组合。

例 1: 把 AL 中的非压缩 BCD 码变成相应十进制数的 ASCII 码。

```
OR AL, 30H
```

#### (3) 逻辑非 NOT

对操作数进行按位逻辑“非”操作。格式: NOT mem/reg

例: NOT CX

```
NOT BYTE PTR[DI]
```

#### (4) 逻辑异或 XOR

对两个操作数按位进行“异或”操作。

格式: XOR dest, src

用途: 对 reg 清零(自身异或)

把 reg/mem 的某几位变反(与' 1' 异或)

例 1: 把 AX 寄存器清零。

```
①MOV AX, 0
```

```
②XOR AX, AX
```

```
③AND AX, 0
```

④SUB AX, AX

#### (5) 测试指令 TEST

操作与 AND 指令类似,但不将”与”的结果送回,只影响标志位。

TEST 指令常用于位测试,与条件转移指令一起用。

例: 测试 AL 的内容是否为负数。

TEST AL,80H ; 检查 AL 中 D<sub>7</sub>=1?

JNZ MINUS ; 是 1(负数), 转 MINUS

... .. ; 否则为正数

## 2、移位指令

### (1)非循环移位指令 (重点🔔)

算术左移指令 SAL(Shift Arithmetic Left)

算术右移指令 SAR(Shift Arithmetic Right)

逻辑左移指令 SHL(Shift Left)

逻辑右移指令 SHR(Shift Right)

这 4 条指令的格式相同,以 SAL 为例:

SAL mem/reg { CL ;移位位数大于 1 时  
1 ;移位位数等于 1 时

➤算术移位——把操作数看做有符号数;

逻辑移位——把操作数看做无符号数。

➤移位位数放在 CL 寄存器中, 如果只移 1 位,也可以直接写在指令中。例如:

MOV CL,4

SHR AL,CL ; AL 中的内容右移 4 位

➤影响 C,P,S,Z,O 标志。

➤结果未溢出时:

左移 1 位 $\equiv$ 操作数 $\times 2$

右移 1 位 $\equiv$ 操作数 $/2$

例: 把 AL 中的数  $x$  乘 10

因为  $10=8+2=2^3+2^1$ , 所以可用移位实现乘 10 操作。程序如下:

MOV CL,3

SAL AL,1 ;  $2x$

MOV AH,AL

SAL AL,1 ;  $4x$

SAL AL,1 ;  $8x$

ADD AL,AH ;  $8x+2x=10x$

## 四、控制转移指令

### 1、转移指令

### (1) 无条件转移指令 JMP

格式: JMP label

本指令无条件转移到指定的目标地址,以执行从该地址开始的程序段。

### (2) 条件转移指令 (补充内容) (重点🔔)

#### ① 根据单个标志位设置的条件转移指令

JB/JC ; 低于,或 CF=1,则转移  
JNB/JNC/JAE ; 高于或等于,或 CF=0,则转移  
JP/JPE ; 奇偶标志 PF=1(偶),则转移  
JNP/JPO ; 奇偶标志 PF=0(奇),则转移  
JZ/JE ; 结果为零(ZF=1),则转移  
JNZ/JNE ; 结果不为零(ZF=0),则转移  
JS ; SF=1,则转移  
JNS ; SF=0,则转移  
JO ; OF=1,则转移  
JNO ; OF=0,则转移

#### ②根据组合条件设置的条件转移指令

这类指令主要用来判断两个数的大小。

判断无符号数的大小

- JA 高于则转移  
条件为:  $CF=0 \wedge ZF=0$ , 即  $A > B$
- JNA/JBE 低于或等于则转移  
条件为:  $CF=1 \vee ZF=1$ , 即  $A \leq B$
- JB  $A < B$  则转移
- JNB  $A \geq B$  则转移

★判断有符号数的大小

- JG ; 大于则转移( $A > B$ )  
条件为:  $(SF \oplus OF=0) \wedge ZF=0$
- JGE; 大于或等于则转移( $A \geq B$ )  
条件为:  $(SF \oplus OF=0) \vee ZF=1$
- JLE; 小于或等于则转移( $A \leq B$ )  
条件为:  $(SF \oplus OF=1) \vee ZF=1$
- JL; 小于则转移( $A < B$ )  
条件为:  $(SF \oplus OF=1) \wedge ZF=0$

## 2、循环控制指令

- 用在循环程序中以确定是否要继续循环。
- 循环次数通常置于 CX 中。
- 转移的目标应在距离本指令-128~+127 的范围之内。
- 循环控制指令不影响标志位。

### (1)LOOP



格式: LOOP label

操作: (CX)-1→CX;

若(CX)≠0,则转至 label 处执行;

否则退出循环,执行 LOOP 后面的指令。

LOOP 指令与下面的指令段等价:

DEC CX

JNZ label

### 3、过程调用指令

#### (1) 调用指令 CALL

一般格式: CALL sub ;sub 为子程序的入口

### 4、中断指令

(1)INT n 执行类型 n 的中断服务程序, N=0~255

## 五、处理器控制指令

### 1、标志位操作

#### (1) CF 设置指令

**CLC** 0→CF

**STC** 1→CF

**CMC** CF 变反

#### (2) DF 设置指令

**CLD** 0→DF (串操作的指针移动方向从低到高)

**STD** 1→DF (串操作的指针移动方向从高到低)

#### (3) IF 设置指令

**CLI** 0→IF (禁止 INTR 中断)

**STI** 1→IF (开放 INTR 中断)

### 2、HLT (halt)

执行 HLT 指令后, CPU 进入暂停状态。

## 第四章 8086 汇编语言程序设计

### 第一节 伪指令 (重点🔔)

CPU 指令与伪指令之间的区别:

(1)CPU 指令是给 CPU 的命令, 在运行时由 CPU 执行, 每条指令对应 CPU 的一种特定的操作。而伪指令是给汇编程序的命令, 在汇编过程中由汇编程序进行处理。

(2)汇编以后, 每条 CPU 指令产生一一对应的目标代码; 而伪指令则不产生与之相应的目标代码。

### 1、数据定义伪指令

#### (1) 数据定义伪指令的一般格式为:

●[变量名] 伪指令 操作数[, 操作数...]

**DB** 用来定义字节 (BYTE)

**DW** 用来定义字 (WORD)

**DD** 用来定义双字 (DWORD)

(2) 操作数的类型可以是:

①常数或常数表达式

●例如: DATA\_BYTE DB 10,5,10H  
DATA\_WORD DW 100H,100,-4  
DATA\_DW DD 2\*30,0FFFBH

☞可以为字符串 (定义字符串最好使用 DB)

●例如: char1 DB 'AB'

Ω可以为变量

Ⅲ可以为? 号操作符

例如: X DB 5, ?, 6

? 号只是为了给变量保留相应的存储单元, 而不赋予变量某个确定的初值。

△重复次数: N DUP (初值[, 初值...])

●例如: ZERO DB 2 DUP (3, 5)

XYZ DB 2 DUP (0, 2 DUP (1, 3), 5)

Ⅳ在伪操作的操作数字段中若使用\$,则表示的是地址计数器的当前值。

2、补充内容:

(1) 类型 PTR 地址表达式例如: MOV BYTE PTR [BX], 12H

INC BYTE PTR [BX]

注意: 单操作数指令, 当操作数为基址、变址、基+变的时候必须定义

3、符号定义伪指令

(1) EQU

格式: 名字 EQU 表达式

EQU 伪指令将表达式的值赋予一个名字, 以后可用这个名字来代替上述表达式。

例: CONSTANT EQU 100

NEW\_PORT EQU PORT\_VAL+1

(2) = (等号)

与 EQU 类似, 但允许重新定义

例:

⋮

EMP=7 ; 值为 7

⋮

EMP=EMP+1 ; 值为 8

(3) LABEL

LABEL 伪指令的用途是定义标号或变量的类型

格式: 名字 LABEL 类型

变量的类型可以是 BYTE, WORD, DWORD。标号的类型可以是 NEAR 或 FAR

4、段定义伪指令

与段有关的伪指令有:

## SEGMENT、ENDS、ASSUME、ORG

(1) 段定义伪指令的格式如下：

```
段名 SEGMENT [定位类型] [组合类型] ['类别']  
:  
段名 ENDS
```

### SEGMENT 和 ENDS

这两个伪指令总是成对出现，二者前面的段名一致。二者之间的删节部分，对数据段、附加段及堆栈段，一般是符号、变量定义等伪指令。对于代码段则是指令及伪指令。此外，还必须明确段和段寄存器的关系，这可由 ASSUME 语句来实现。

(2) ASSUME

格式：

ASSUME 段寄存器名：段名[，段寄存器名：段名[，...]]

ASSUME 伪指令告诉汇编程序，将某一个段寄存器设置为某一个逻辑段址，即明确指出源程序中逻辑段与物理段之间的关系。

(3) ORG

伪指令 ORG 规定了段内的起始地址或偏移地址，其格式为：

ORG <表达式>

表达式的值即为段内的起始地址或偏移地址，从此地址起连续存放程序或数据。

## 5、汇编程序的一般结构 (重点🔔) (记住)

```
DATA SEGMENT  
...  
DATA ENDS  
CODE SEGMENT  
ASSUME CS:CODE,DS:DATA  
BGN: MOV AX,DATA  
      MOV DS,AX  
      ....  
      MOV AH,4CH  
      INT 21H  
CODE ENDS  
END BGN
```

## 第三节 程序设计

1、顺序程序的设计 (略)

2、分支程序的设计

典型例题：

$$Y = \begin{cases} 1 & X > 0 \\ 0 & X = 0 \\ -1 & X < 0 \end{cases}$$

●程序为:

```
MOV AL, X
CMP AL, 0
JGE BIG
MOV Y, -1
JMP EXIT
BIG: JE EQU
MOV Y, 1
JMP EXIT
```

```
EQU: MOV Y, 0
EXIT: ....
```

数制 { 二进制数(B)  
八进制数(Q)  
十六进制数(H)  
十进制数(D)

3、循环程序见讲义。

●用计数控制循环

第一章 计算机基础知识

本章的主要内容包括: 数制、数制转换、带符号数编码、原码、反码、补码、数在计算机中的表示、字符在计算机中的表示、奇偶校验码、ASCII码、BCD码、数字编码规则、字母编码规则、压缩BCD码、非压缩BCD码、本章的知识要点图, 图 1.2 为计算机系统组成。

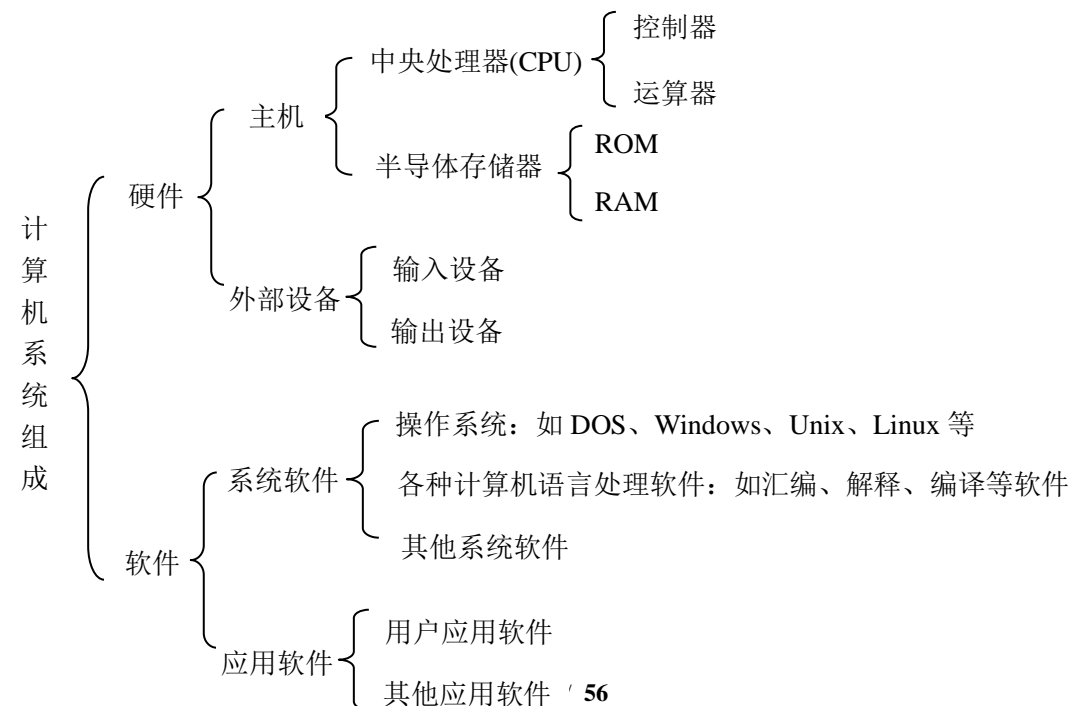
本章的知识要点图, 图 1.2 为计算机系统组成。

知识要点

奇偶校验码 { 奇校验码  
偶校验码

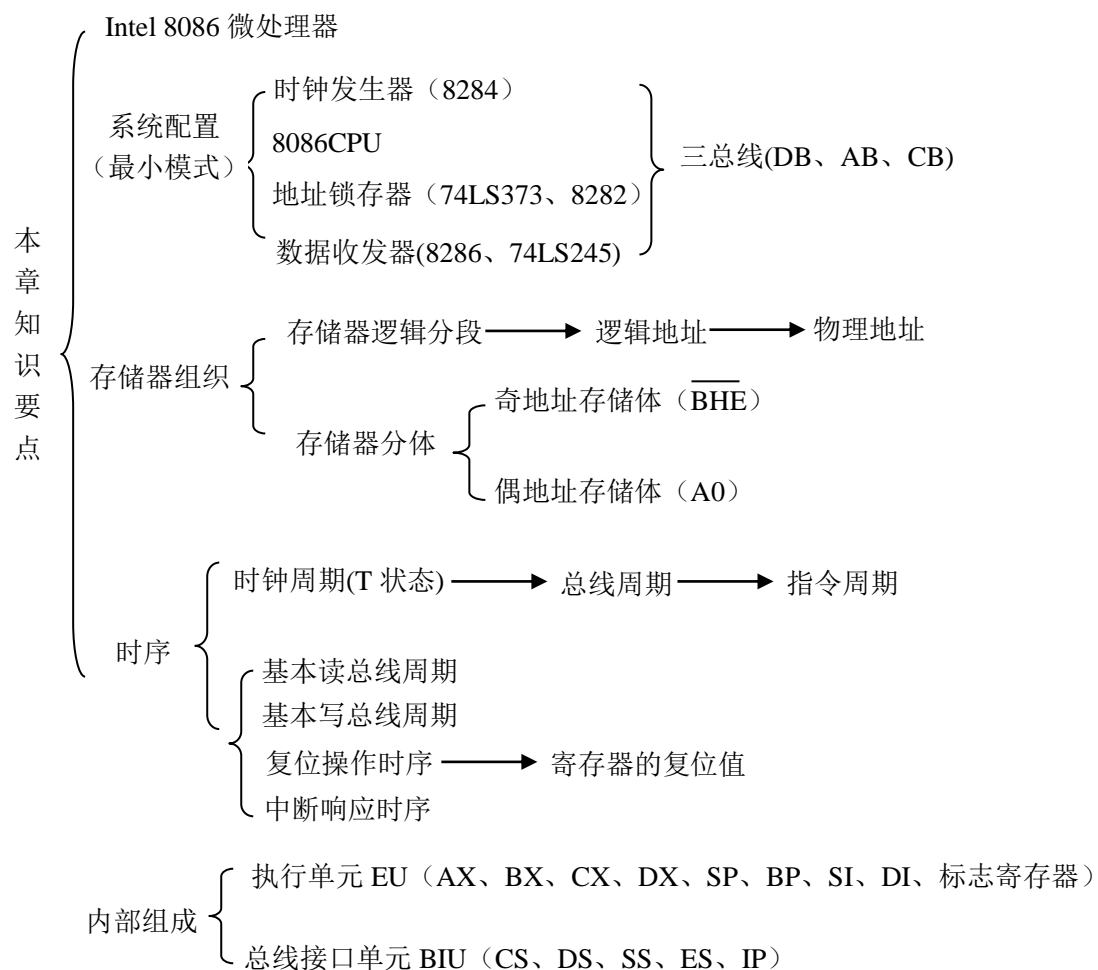
字符编码 { ASCII 码 { 数字编码规则  
字母编码规则  
BCD 码 { 压缩 BCD 码  
非压缩 BCD 码

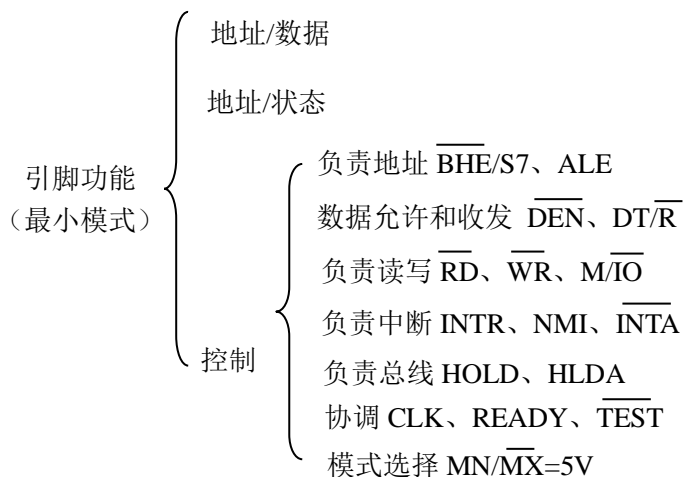
计算机系统组成



## 第二章 8086 微处理器

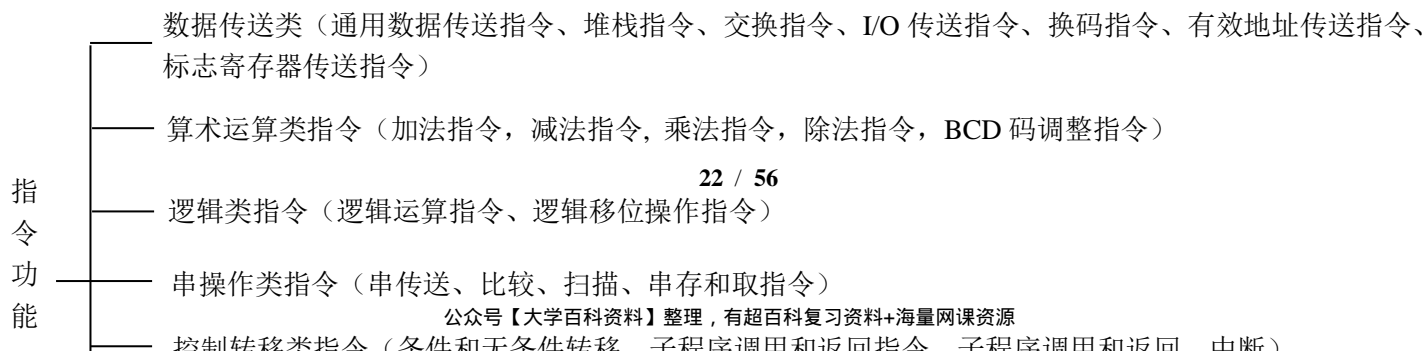
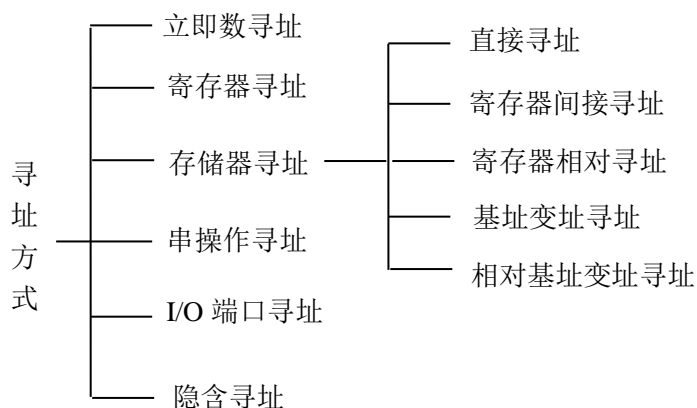
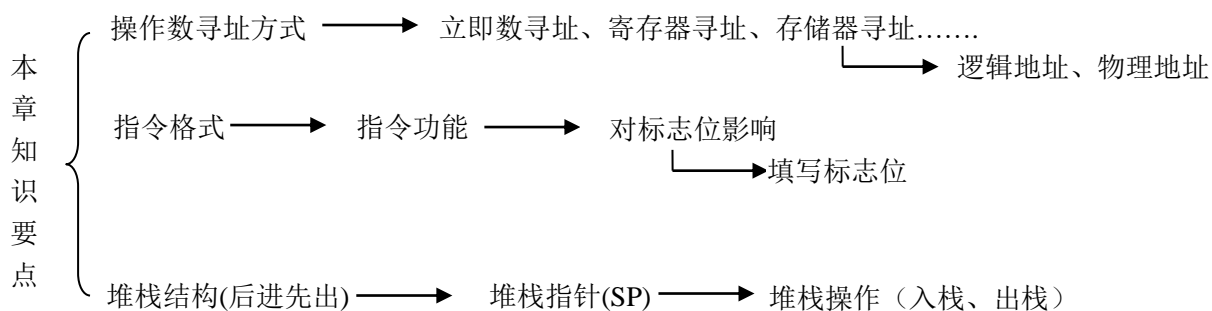
本章要从应用角度上理解 8086CPU 的内部组成、编程结构、引脚信号功能、最小工作模式的系统配置、8086 的存储器组织、基本时序等概念。下面这一章知识的结构图。





### 第三章 8086 的指令系统

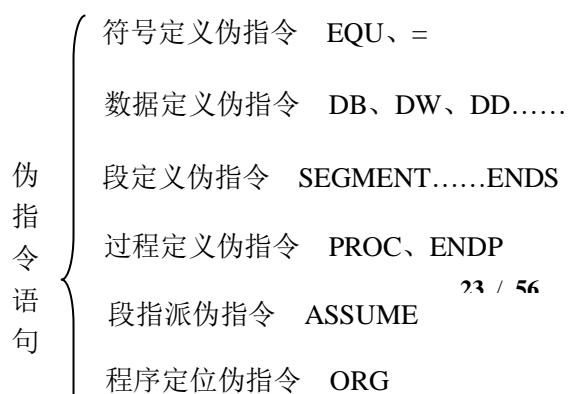
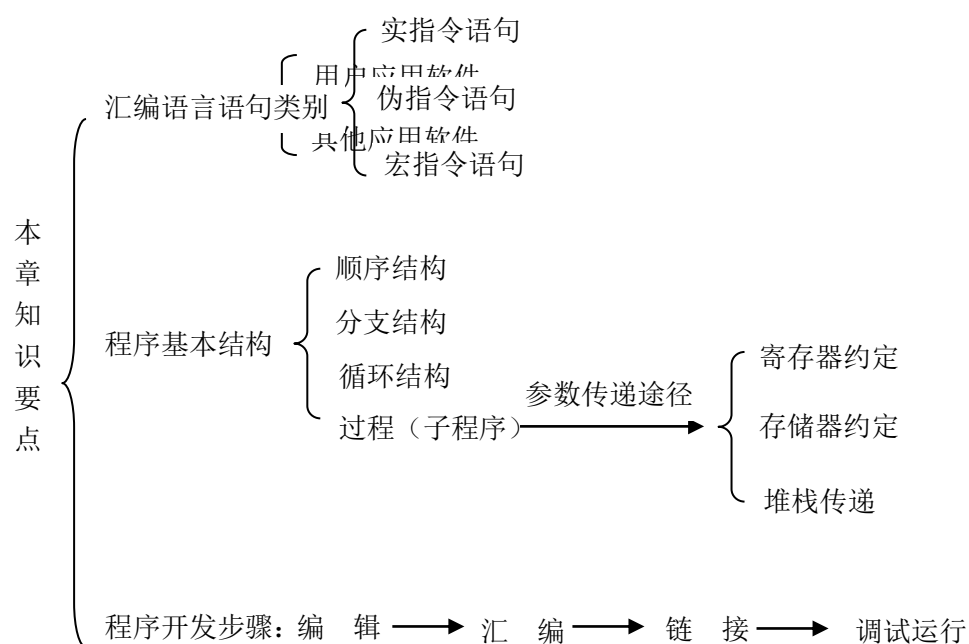
本章重点是 8086CPU 指令的寻址方式，每条指令的格式、功能及标志的影响；同时还涉及到存储器单元的物理地址计算、标志位填写和堆栈操作。下图为本章知识结构图。



## 第四章 汇编语言程序设计

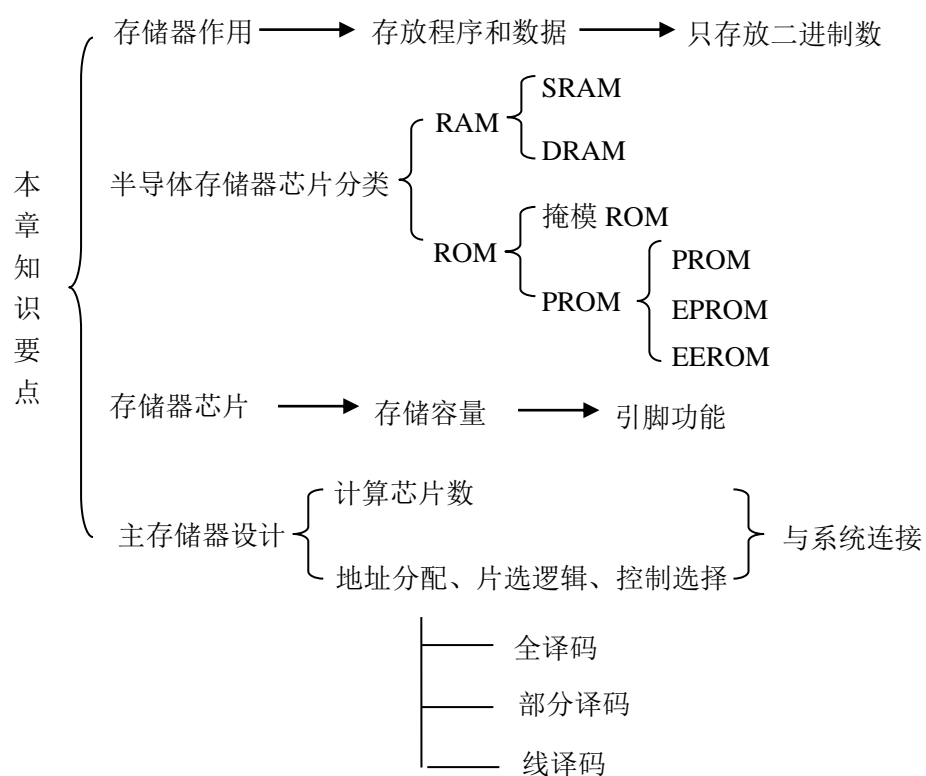
本章主要内容是汇编语言类别、伪指令语句格式和作用、基本程序结构、调用程序和被调用程序之间数据传递途径以及汇编源程序上机调试过程。

本章重点是阅读程序和编写程序。下边是本章的知识结构图。



## 第五章 半导体存储器

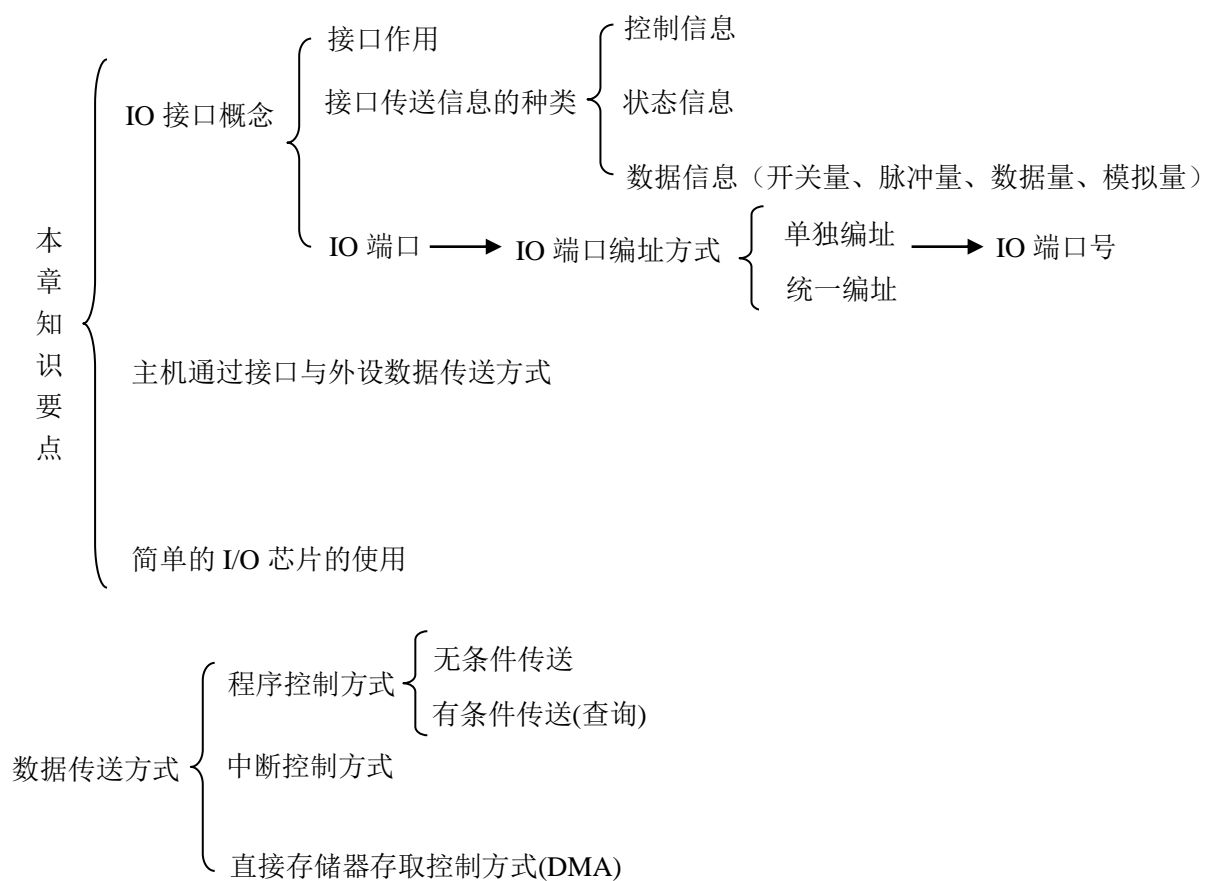
半导体存储器是用半导体器件作为存储介质的存储器。本章讨论半导体存储器芯片的类型、存储原理、引脚功能、如何与 CPU（或系统总线）连接等问题。本章知识结构图如下。





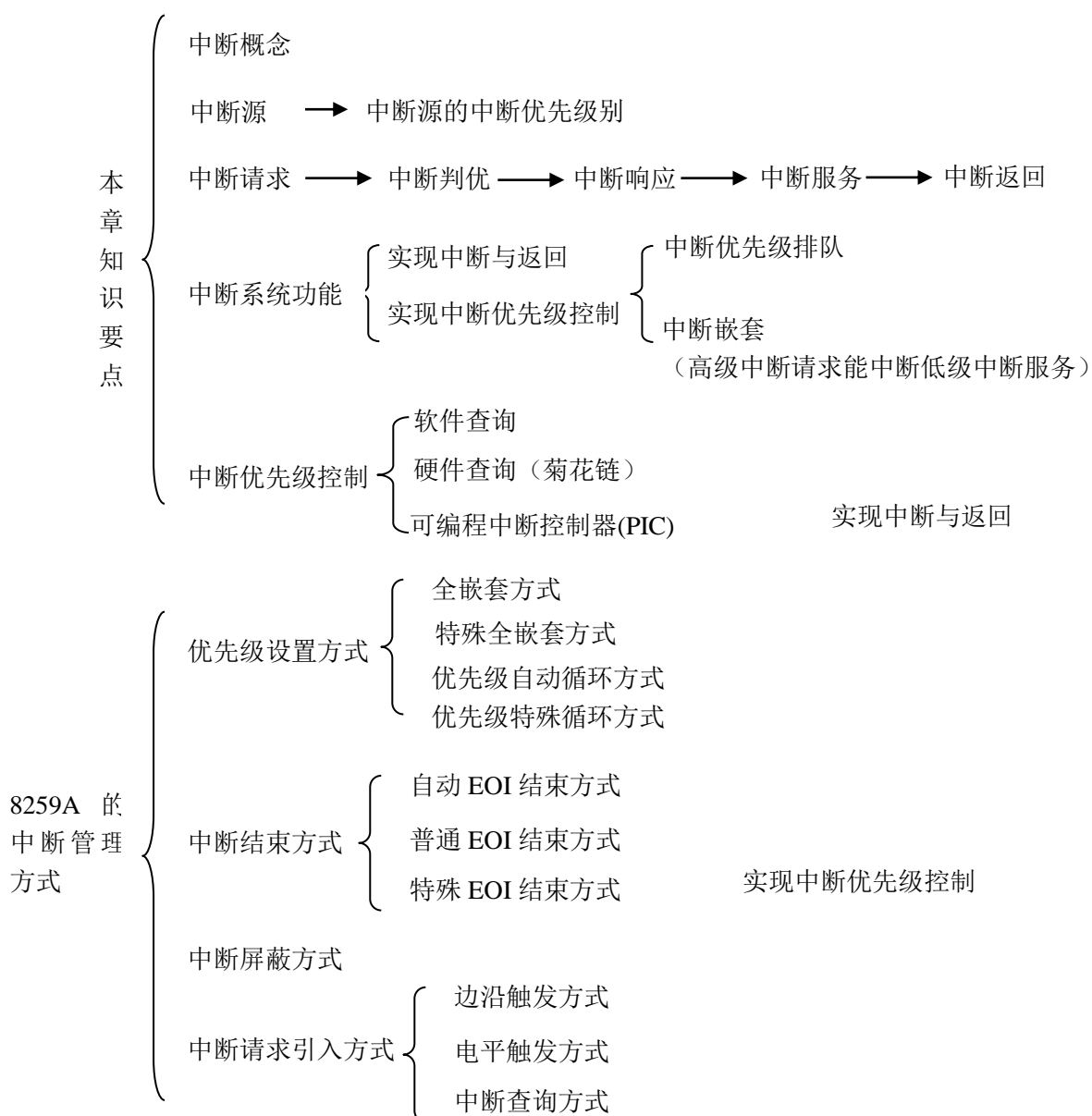
## 第六章 输入输出接口

本章讨论输入/输出接口的基本概念，包括输入/输出接口的作用、内部结构、传送信息的分析、IO 端口编址以及主机通过接口与外设之间数据传送的方式。下边是本章的知识结构图。



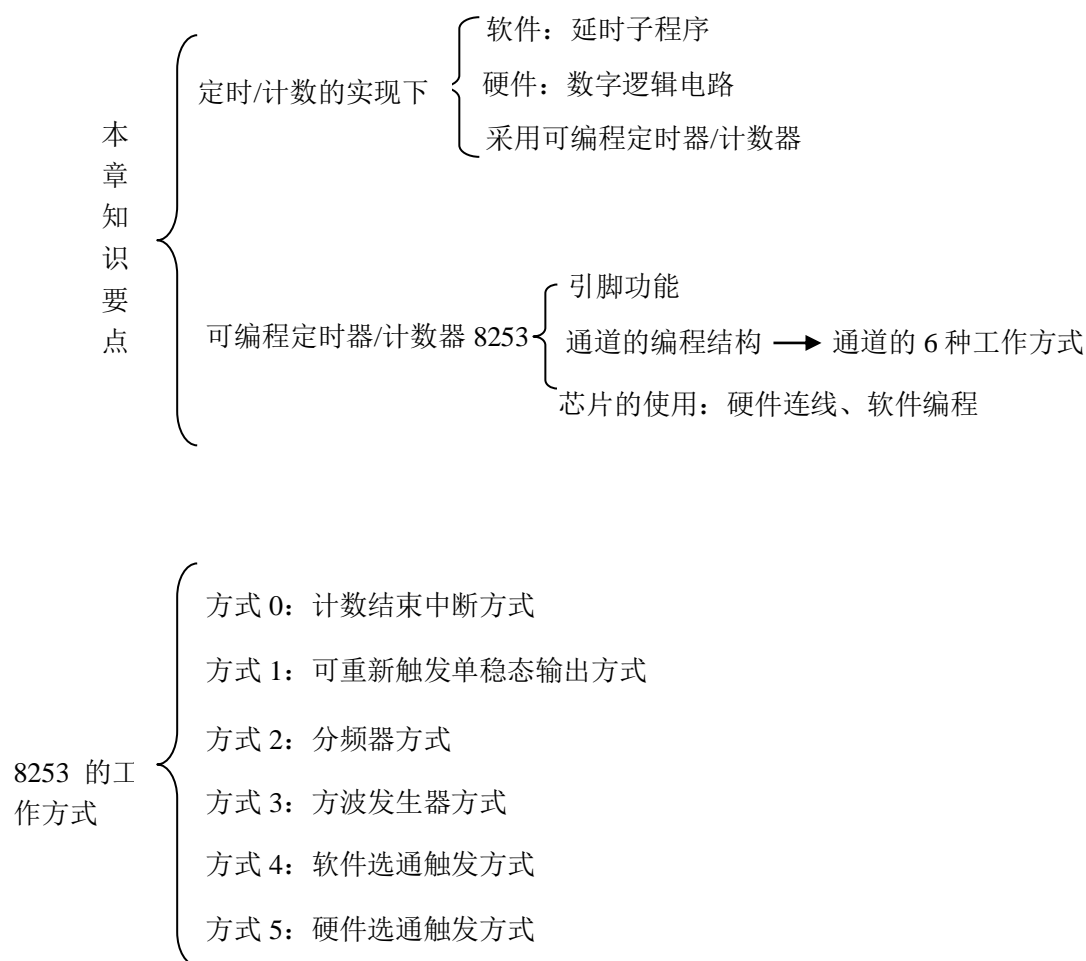
## 第七章 中断与中断控制器

本章主要内容：中断的基本概念、CPU 响应中断的条件、中断响应过程、中断服务程序的执行；8086/8088 中断系统；可编程中断控制器 8259A 的引脚功能、编程结构以及工作工程。



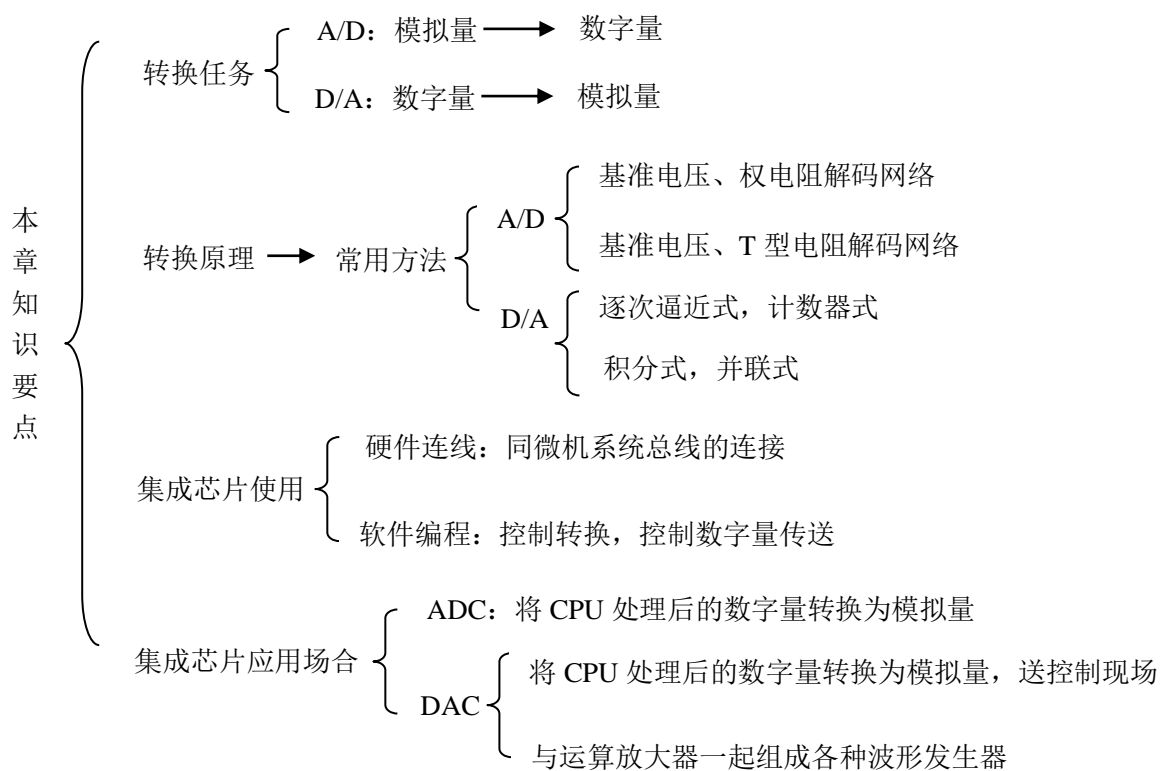
## 第八章 定时器/计数器 8253 及应用

本章主要内容是定时器/计数器的应用场合；如何实现定时/计数；可编程计数器/定时器 8253 芯片的内部结构、引脚功能、计数原理、6 种工作方式下的工作条件和输出波形特征。下边是知识要点图。



## 第九章 A/D 和 D/A 转换

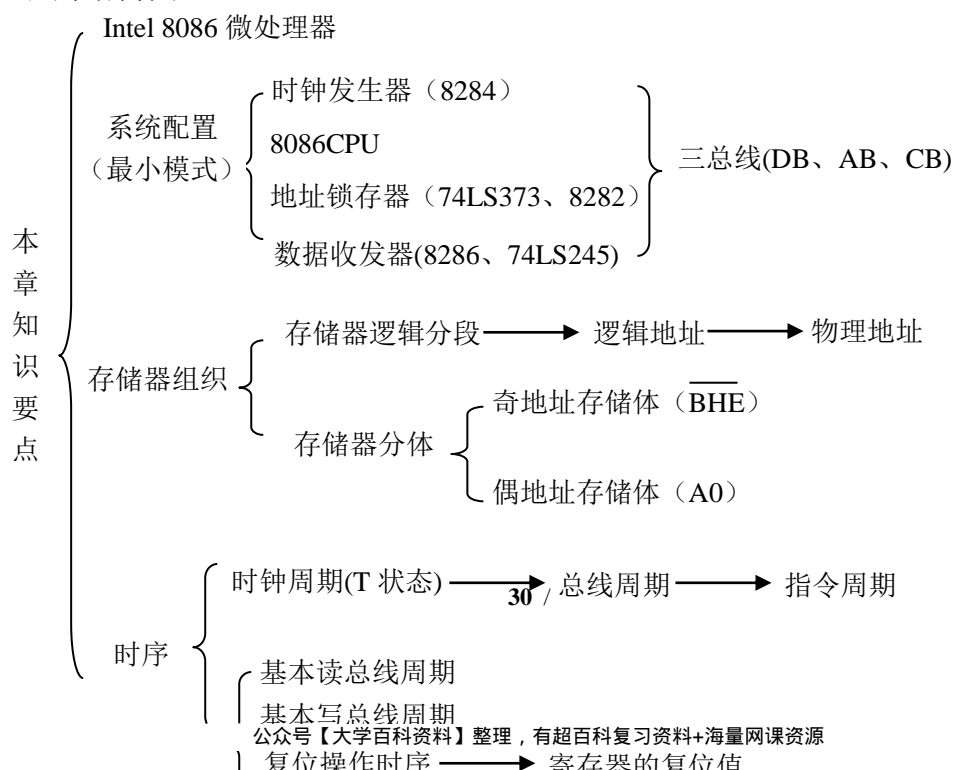
本章重点是 A/D 转换的任务和转换原理, D/A 转换的任务和转换原理, 常用 A/D 转换器(ADC)集成芯片和 D/A 转换器(DAC)集成芯片的外部引脚功能、内部结构、工作过程、性能指标以及实际应用。

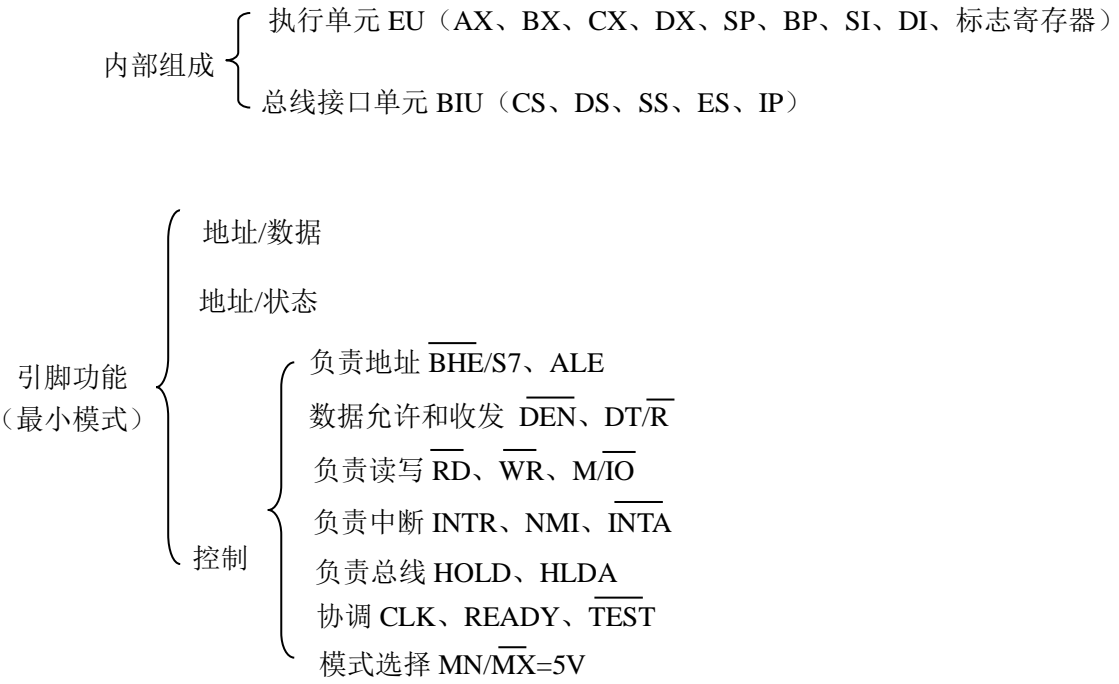




## 第二章 8086 微处理器

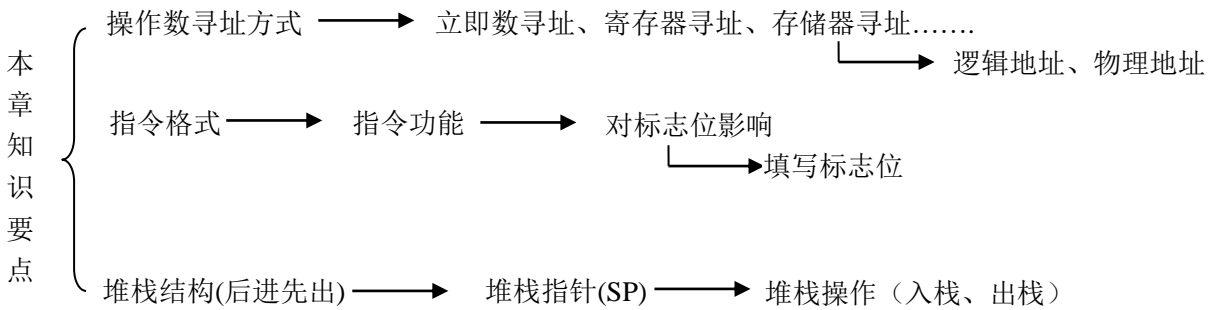
本章要从应用角度上理解 8086CPU 的内部组成、编程结构、引脚信号功能、最小工作模式的系统配置、8086 的存储器组织、基本时序等概念。下面这一章知识的结构图。

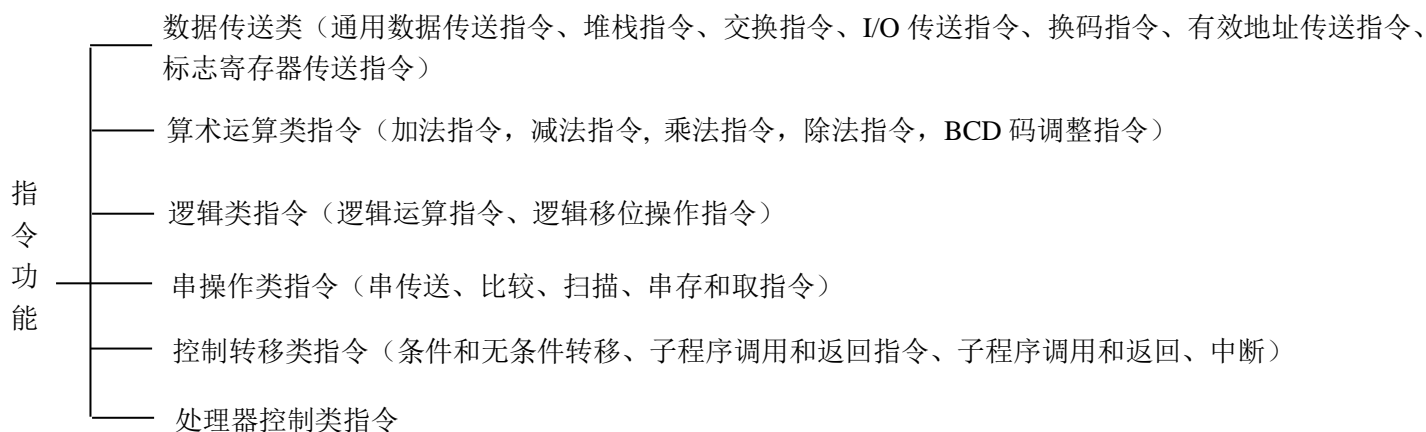
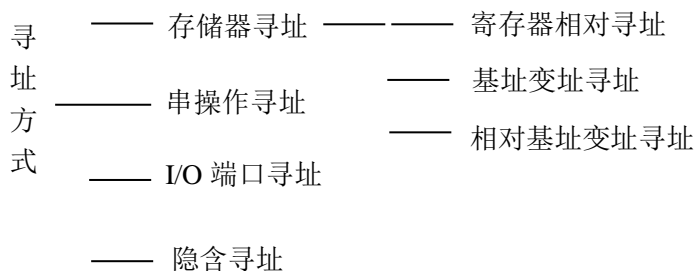




### 第三章 8086 的指令系统

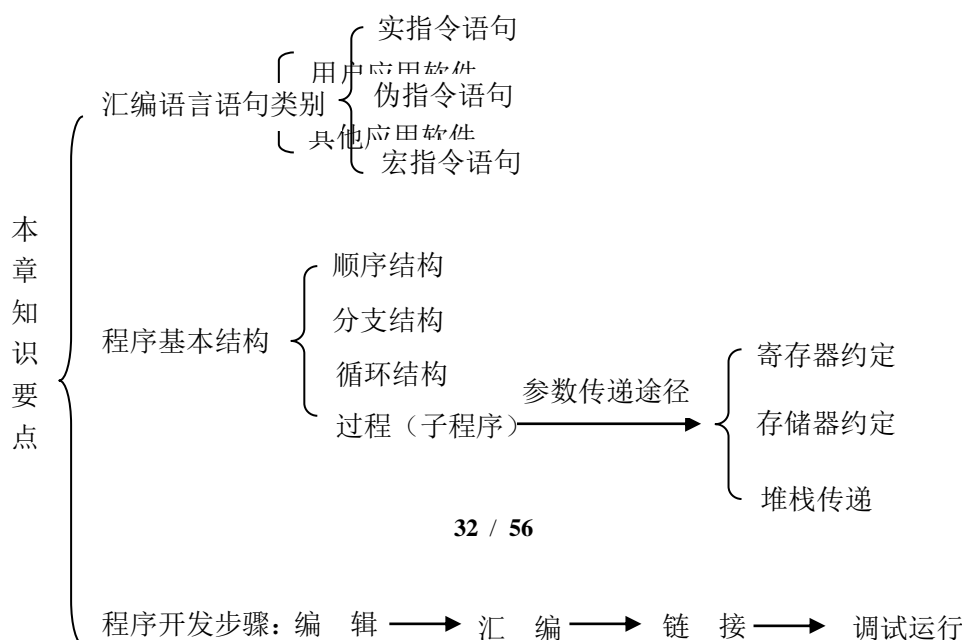
本章重点是 8086CPU 指令的寻址方式，每条指令的格式、功能及标志的影响；同时还涉及到存储器单元的物理地址计算、标志位填写和堆栈操作。下图为本章知识结构图。





#### 第四章 汇编语言程序设计

本章主要内容是汇编语言类别、伪指令语句格式和作用、基本程序结构、调用程序和被调用程序之间数据传递途径以及汇编源程序上机调试过程。本章重点是阅读程序和编写程序。下边是本章的知识结构图。

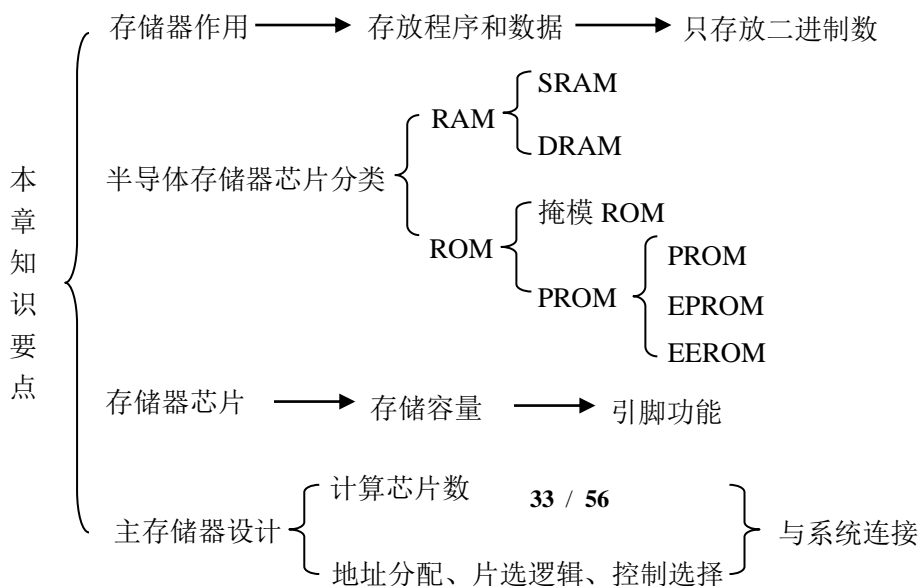






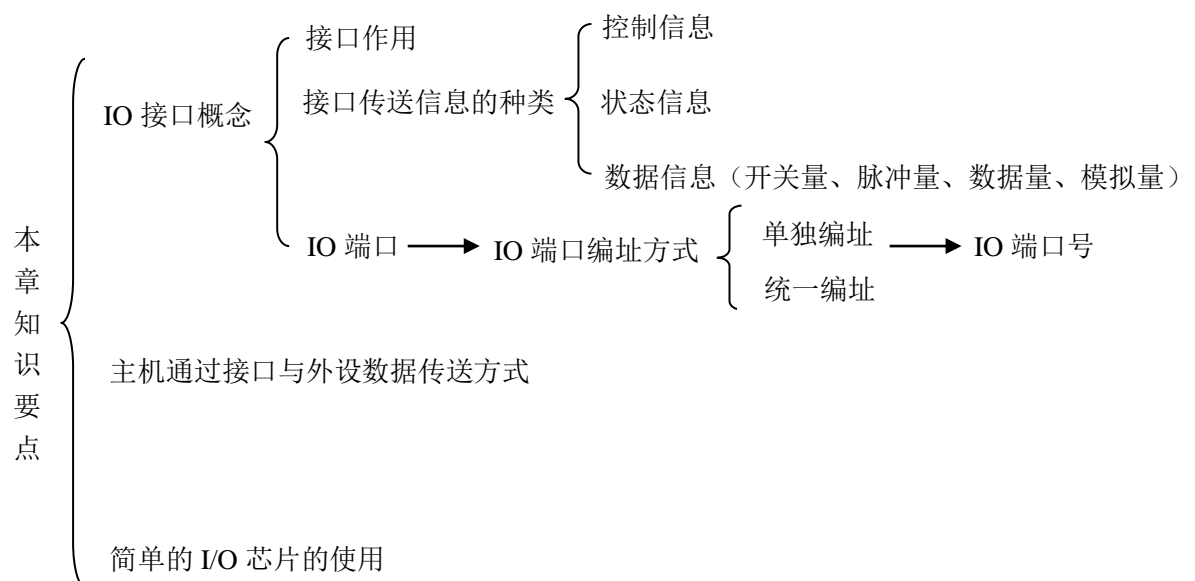
## 第五章 半导体存储器

半导体存储器是用半导体器件作为存储介质的存储器。本章讨论半导体存储器芯片的类型、存储原理、引脚功能、如何与 CPU（或系统总线）连接等问题。本章知识结构图如下。



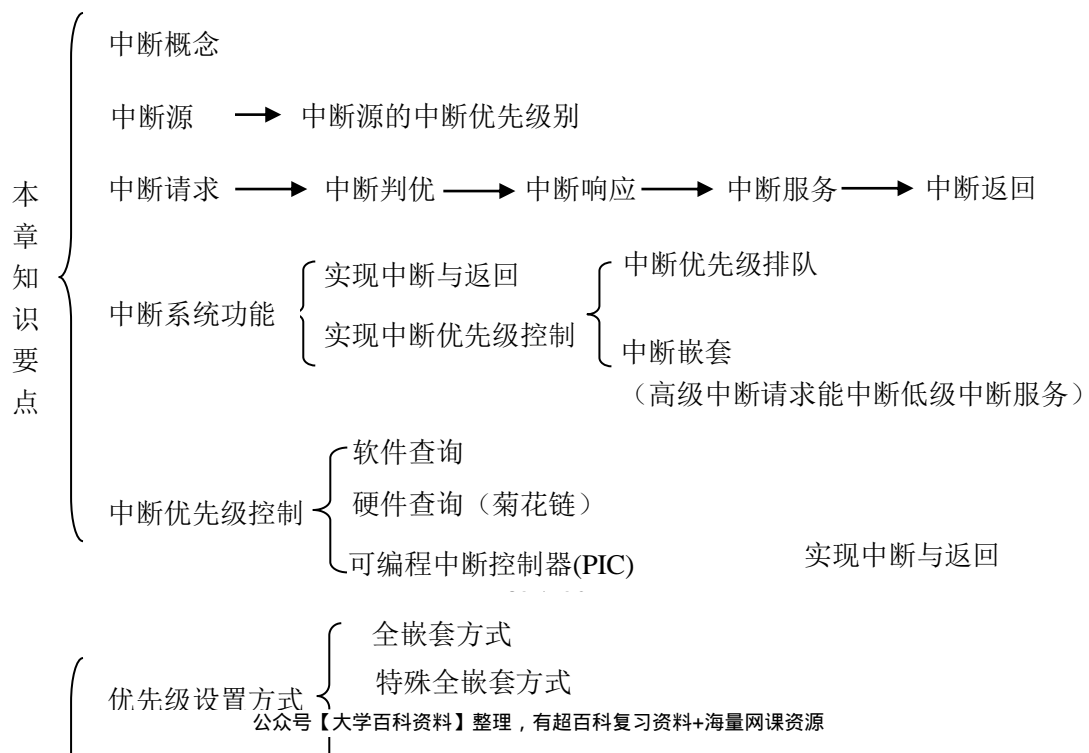
## 第六章 输入输出接口

本章讨论输入/输出接口的基本概念，包括输入/输出接口的作用、内部结构、传送信息的分析、IO 端口编址以及主机通过接口与外设之间数据传送的方式。下边是本章的知识结构图。



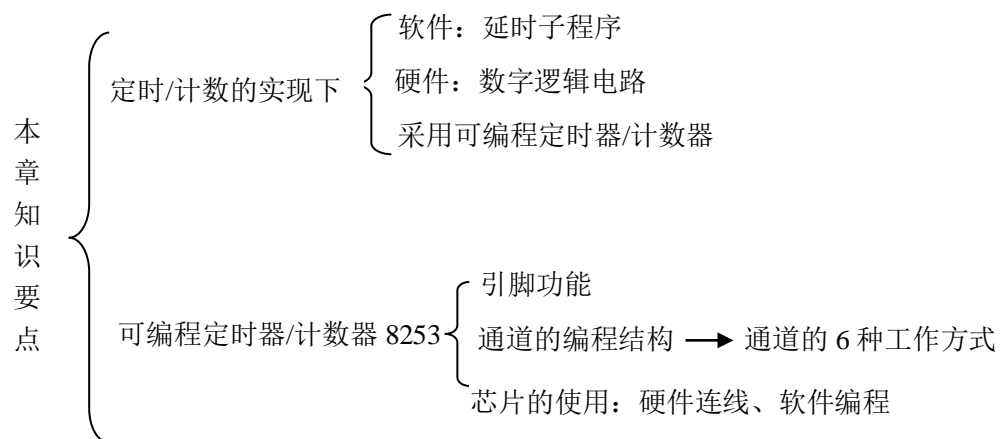
## 第七章 中断与中断控制器

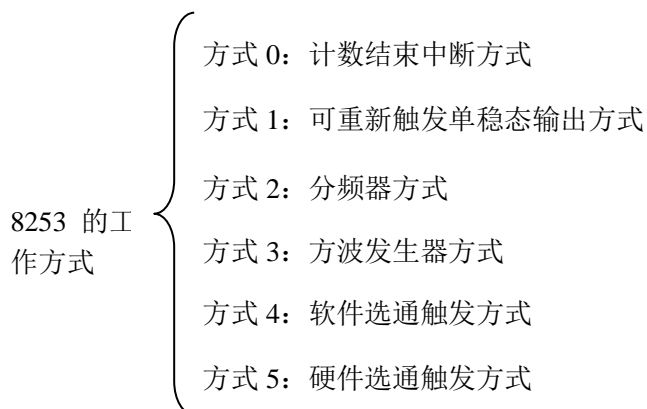
本章主要内容：中断的基本概念、CPU 响应中断的条件、中断响应过程、中断服务程序的执行；8086/8088 中断系统；可编程中断控制器 8259A 的引脚功能、编程结构以及工作工程。



## 第八章 定时器/计数器 8253 及应用

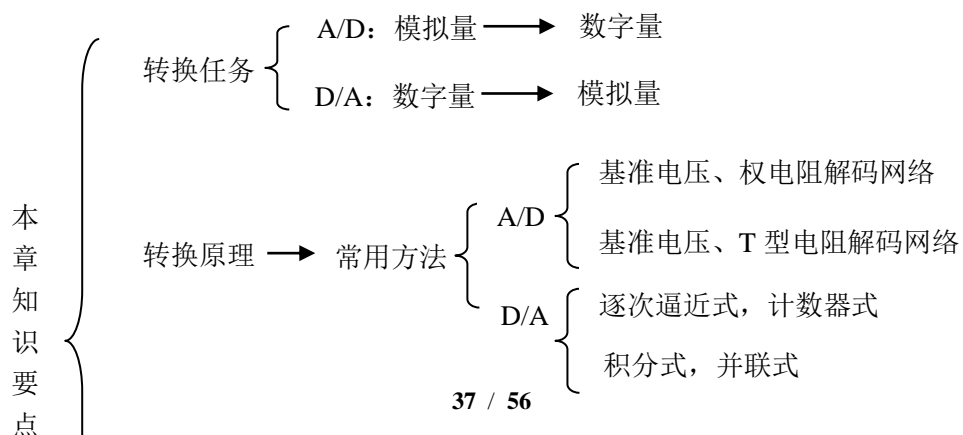
本章主要内容是定时器/计数器的应用场合；如何实现定时/计数；可编程计数器/定时器 8253 芯片的内部结构、引脚功能、计数原理、6 种工作方式下的工作条件和输出波形特征。下边是知识要点图。

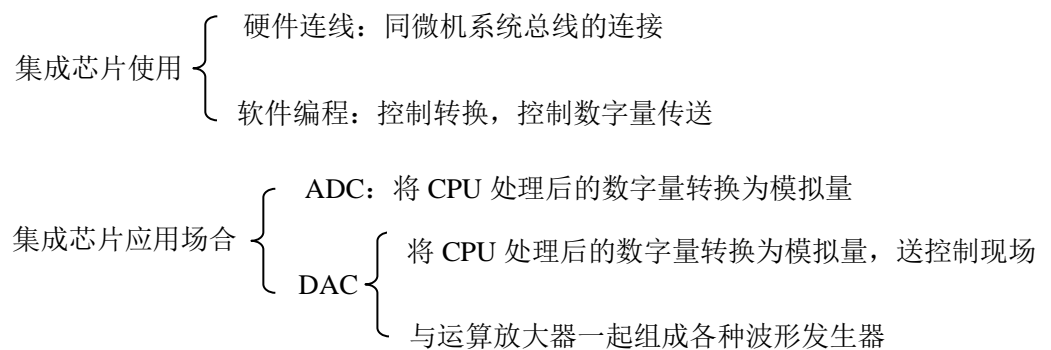




## 第九章 A/D 和 D/A 转换

本章重点是 A/D 转换的任务和转换原理, D/A 转换的任务和转换原理, 常用 A/D 转换器(ADC) 集成芯片和 D/A 转换器(DAC)集成芯片的外部引脚功能、内部结构、工作过程、性能指标以及实际应用。





如下图所示，以 8088 微处理器为核心的 IBM PC/XT 机与 DAC0832 连接，实现波形发生器。IBM PC/XT 机使用 10 根地址线 A0~A9 寻址 I/O 端口，AEN 为地址允许信号，低电平时选中端口。DAC0832 的参考电压  $V_{REF} = -5V$ ， $V_{REF}$  的范围为 0~5V，计算式为

$$V_{out1} = -\left(\frac{N}{255}\right)V_{REF}, \text{ 其中, } N \text{ 是由 DAC0832 转换的数字量对应的十进制值。} V_{out} \text{ 的输出}$$

范围是 -5V~5V。

- (1) 根据下图一所示的 DAC0832 的硬件连接，说明其工作方式。
- (2) 假如 DAC0832 端口地址为 140H，请在下图一中画出相应的译码电路。
- (3) 现有 1ms 的延时子程序 DELAY，请编写程序片段实现输出右图二的所示波形。

## 1. 什么是中断？什么是中断向量？中断向量的地址范围？

答：中断就是 CPU 在执行当前程序时由于内外部事件引起 CPU 暂时停止当前正在执行的程序而转向执行请求 CPU 暂时停止的内外部事件的服务程序，该程序处理完后又返回继续执行被停止的程序；**中断向量是中断处理子程序的入口地址；地址范围是 00000H-003FFH。**

2.

3. 微机系统的硬件由哪几部分组成？

答：微型计算机（微处理器，存储器，I/O 接口，系统总线），外围设备，电源。

4. 什么是微机的总线，分为哪三组？

答：是传递信息的一组公用导线。分三组：地址总线，数据总线，控制总线。

5. 8086/8088CPU 的内部结构分为哪两大模块，各自的主要功能是什么？

答：总线接口部件（BIU）功能：根据执行单元 EU 的请求完成 CPU 与存储器或 IO 设备之间的数据传送。执行部件（EU），作用：从指令队列中取出指令，对指令进行译码，发出相应的传送数据或算术的控制信号接受由总线接口部件传送来的数据或把数据传送到总线接口部件进行算术运算。

6. 8086 指令队列的作用是什么？

答：作用是：在执行指令的同时从内存中取了一条指令或下几条指令，取来的指令放在指令队列中这样它就不需要象以往的计算机那样让 CPU 轮番进行取指和执行的工作，从而提高 CPU 的利用率。

7. 8086 的存储器空间最大可以为多少？怎样用 16 位寄存器实现对 20 位地址的寻址？完成逻辑地址到物理地址转换的部件是什么？

答：8086 的存储器空间最大可以为  $2^{20}$ （1MB）；8086 计算机引入了分段管理机制，当 CPU 寻址某个存储单元时，先将段寄存器内的内容左移 4 位，然后加上指令中提供的 16 位偏移地址形成 20 位物理地址。

8. 段寄存器 CS=1200H，指令指针寄存器 IP=FF00H，此时，指令的物理地址为多少？指向这一物理地址的 CS 值和 IP 值是唯一的吗？

答：指令的物理地址为 21F00H；CS 值和 IP 值不是唯一的，例如：CS=2100H，IP=0F00H。

9. 设存储器的段地址是 4ABFH，物理地址为 50000H，其偏移地址为多少？

答：偏移地址为 54100H。（物理地址=段地址\*16+偏移地址）

10. 8086/8088CPU 有哪几个状态标志位，有哪几个控制标志位？其意义各是什么？

答：状态标志位有 6 个：ZF，SF，CF，OF，AF，PF。其意思是用来反映指令执行的特征，通常是由 CPU 根据指令执行结果自动设置的；控制标志位有 3 个：DF，IF，TF。它是由程序通过执行特定的指令来设置的，以控制指令的操作方式。

11. 8086CPU 的 AD0~AD15 是什么引脚？

答：数据与地址引脚

12. INTR、INTA、NMI、ALE、HOLD、HLDA 引脚的名称各是什么？

答：INTR 是可屏蔽请求信号，INTA 中断响应信号，NMI 是不可屏蔽中断请求信号，ALE 是地址锁存允许信号，HOLD 总线请求信号，HLDA 总线请求响应信号。

13. 虚拟存储器有哪两部分组成？

答：有主存储器和辅助存储器。

14. 在 80x86 中，什么是逻辑地址、线性地址、物理地址？

答：线性地址是连续的不分段的地址；逻辑地址是由程序提供的地址；物理地址是内存单元的实际地址。

15. 段描述符分为哪几种？

答：分为三大类，程序段描述符，系统段描述符，门描述符。

**16. RAM 有几种，各有什么特点？ROM 有几种，各有什么特点？**

答：RAM 有两种，SRAM(静态 RAM)，它采用触发器电路构成一个二进制位信息的存储单元，这种触发器一般由 6 个晶体管组成，它读出采用单边读出的原理，写入采用双边写入原理；DRAM（动态 RAM），它集成度高，内部存储单元按矩阵形式排列成存储体，通常采用行，列地址复合选择寻址法。ROM 有 5 种，固定掩模编程 ROM，可编程 PROM，紫外光擦除可编程 EPROM，电可擦除的可编程 EPROM，闪速存储器。

**17. 若用 4K×1 位的 RAM 芯片组成 8K×8 为的存储器，需要多少芯片？**

**A19—A0 地址线中哪些参与片内寻址，哪些用做芯片组的片选信号？**

答：需要 16 片芯片；其中 A11-A0 参与片内寻址；A12 做芯片组的片选信号。

**18. 若系统分别使用 512K×8、1 K×4、16K×8、64K×1 的 RAM，各需要多少条地址线进行寻址，各需要多少条数据线？**

答：512K×8 需要 19 条地址线，8 条数据线。1 K×4 需要 10 条地址线，4 条数据线。16K×8 需要 14 条地址线，8 条数据线。64K×1 需要 14 条地址线，1 条数据线。

**19. 某微机系统的 RAM 容量为 8K×8，若首地址为 4800H，则最后一个单元的地址是多少？**

答：最后一个单元的地址是：4800H+2<sup>13</sup>-1

**20. 什么是总线，微机中的总线通常分为哪几类？**

答：是一组信号线的集合，是一种在各模块间传送信息的公共通路；有四类，片内总线，微处理器总线，系统总线，外总线。

**21. 微处理器为什么需要用接口和外设相连接？**

答：因为许多接口设备中，在工作原理，驱动方式，信息格式以及工作速度方面彼此相差很大，因此为了进行速度和工作方式的匹配，并协助完成二者之间数据传送控制任务。

**22. 一般的 I/O 接口电路有哪四种寄存器，它们各自的作用是什么？**

答：数据输入寄存器，数据输出寄存器，状态寄存器和控制寄存器。数据端口能对传送数据提供缓冲，隔离，寄存的作用；状态寄存器用来保存外设或接口的状态；控制寄存器用来寄存 CPU 通过数据总线发来的命令。

**23. 8086 最多可有多少级中断？按照产生中断的方法分为哪两大类？**

答：有 8 级；按照产生中断的方法可分为硬件中断和软件中断。

**24. 什么是中断？什么是中断向量？中断向量表的地址范围？**

答：中断就是 CPU 在执行当前程序时由于内外部事件引起 CPU 暂时停止当前正在执行的程序而转向执行请求 CPU 暂时停止的内外部事件的服务程序，该程序处理完后又返回继续执行被停止的程序；中断向量是中断处理子程序的入口地址；地址范围是 00000H-003FFH。

**25. 中断向量表的功能是什么？若中断向量号分别为 1AH 和 20H，则它们的中断向量在中断向量表的什么位置上？**

答：中断向量表的功能是当中断源发出中断请求时，即可查找该表，找出其中断向量，就可转入相应的中断服务子程序。1AH 在中断向量表的位置是 1AH\*4=68H 在中断向量表 0000: 0068 处；20H 在中断向量表的位置是 80H 在中断向量表 0000: 0080 处。

**26. 通常，解决中断优先级的方法有哪几种？**



答：3 种，软件查询确定优先级，硬件优先级排队电路确定优先级，具体中断屏蔽的接口电路。

27. 8259A 通过级联的方式可以由几片构成最多多少级优先权的中断源。

答：8259A 通过级联的方式由 9 片构成最多 64 级优先权的中断源。

28. 简述中断控制器 8259A 的内部结构和主要功能。

答：8259A 的内部结构有数据总线缓冲器，读写逻辑电路，级联缓冲比较器，中断请求寄存器（IRR），中断屏蔽寄存器（IMR），中断服务寄存器（ISR），优先级判别器（PR），控制逻辑。

29. 8259A 的内部寄存器中 IRR、IMR、ISR 三个寄存器的作用是什么？

答：见课本 153 页。

30. 8259A 有哪些中断结束方式，分别适用于哪些场合。

答：8259A 有 2 种中断结束方式：中断自动结束方式，中断非自动结束方式（一般中断和特殊中断）；中断自动结束方式只适合有一块 8259A，并且各中断不发生嵌套的情况。中断非自动结束方式只能适合与全嵌套方式下不能用与循环优先级方式。

31. 8259A 对优先级的管理方式有哪几种，各是什么含义？

答：有 4 种，普通全嵌套方式，特殊全嵌套方式，自动循环方式，优先级特殊循环方式（详细见课本 P159 和 P160）

32. 8259A 的初始化命令字和操作命令字有哪些，其功能是什么；哪些应写入奇地址，哪些应写入偶地址。

答：8259A 的初始化命令字 ICW1, ICW2, ICW3, ICW4；操作命令字 OCW1, OCW2, OCW3。（见课本 P155 到 P158）；ICW2, ICW3, ICW4, OCW1 写如奇地址，ICW1, OCW2, OCW3 为偶地址。

33. 简述 8259A 的初始化过程。

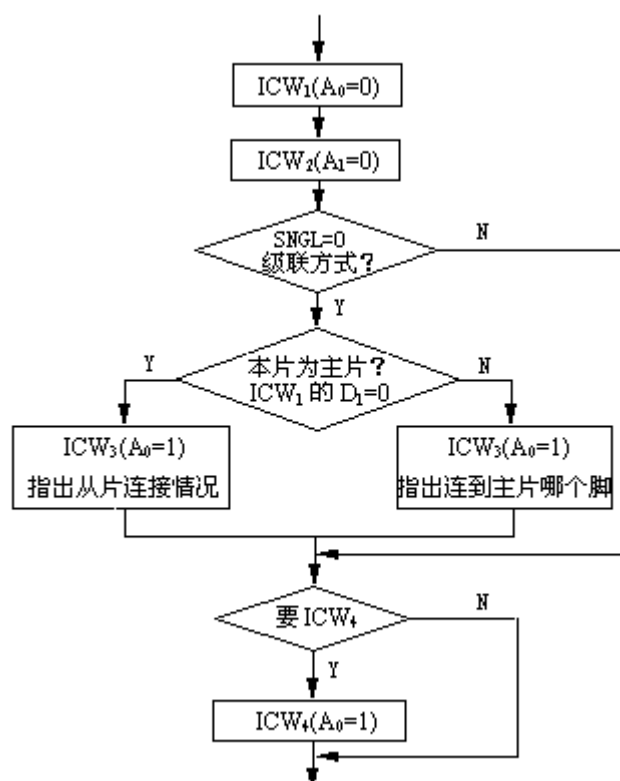
答：8259A 的初始化编程，需要 CPU 向它输出一个 2—4 字节的初始化命令字，输出初始化命令字的流程如图所示，其中 ICW<sub>1</sub> 和 ICW<sub>2</sub> 是必须的，而 ICW<sub>3</sub> 和 ICW<sub>4</sub> 需根据具体的情况来加以选择。各初始化命令字的安排与作用分叙如下：

34. 8253 有几个计数通道，每条计数通道有哪些信号线，其作用是什么？

答：8253 有三个计数通道，每个计数通道有 3 条信号线：CLK：计数输入用于输入定时基准脉冲或计数脉冲。OUT：输出信号以相应的电平指示计数的完成或输出脉冲的波形。GATA：选通输入用于启动或禁止计数器的操作，以使计数器和计数输入信号同步。

35. 8253 有几种工作方式，其特点是什么？

答：六种方式（见课本 P224）



### 36. 8253 的内部寄存器及各位的意义是什么？

答：8253 的内部寄存器有四个，8 位的控制寄存器：初始化时，将控制字写入该寄存器；16 位的计数器初值寄存器，初始化是写入该计数器的初始值，其最大初始值为 0000H；16 位的减一计数器，计数器的初值由计数初值寄存器送入减法计数器，当计数输入端输入一个计数脉冲时，减法计数器内容减一；16 位的输出锁存器用来锁存计数脉冲时，减法计数器内容减一。

### 37. 8255A 的功能是什么，有哪几个控制字，各位的意义是什么？

答：8255A 是一种通用的可编程程序并行 I/O 接口芯片。它有两个控制字，一个是方式选择控制字，它的作用是实现 8255A 的各个端口的选择。一个是对 C 口进行置位或复位控制字。它的作用是能实现对端口 C 的每一位进行控制。

### 38. 8255A 的 A 口、B 口、C 口有哪几种工作方式，其特点是什么？C 口有哪些使用特点？

答：8255A 的 A 口可以工作在 3 种工作方式的任何一种，B 口只能工作在方式 0 或方式 1，C 口则常常配合端口 A 和端口 B 工作，为这两个端口的输入/输出传输提供控制信号和状态信号。

### 39. 同步通信、异步通信的帧格式各是什么？什么是奇、偶校验？

答：异步通信的帧格式是用一个起始位表示传送字符的开始，用 1-2 个停止位表示字符结束。起始位与停止位之间是数据位，数据位后是校验位，数据的最底位紧跟起始位，其他各位顺序传送；同步通信的帧格式是在每组字符之前必须加上一个或多个同步字符做为一个信息帧的起始位。

### 40. 什么是波特率？若在串行通信中的波特率是 1200b/s，8 位数据位，1 个停止位，无校验位，传输 1KB 的文件需要多长时间？

答：波特率是单位时间内通信系统所传送的信息量。

需要多长时间 =  $1024 / (1200/10) = 8.53s$

### 41. 对 8255A 进行初始化，要求端口 A 工作于方式 1，输入；端口 B 工作于方式 0，输出；端口 C 的高 4 位配合端口 A 工作，低 4 位为输入。设控制口的地址为 006CH。

答：由题知应为 10111001H(B9H)

```
MOV AL, B9H
MOV DX, 006CH
OUT DX, AL
```

### 42. 设 8255A 的四个端口地址分别为 00C0H、00C2H、00C4H 和 00C6H，要求用置 0、置 1 的方法对 PC6 置 1，对 PC4 置 0。

答：MOV DX, 00C0H ; 端口地址

MOV AL, 00001101 ; 对 PC6 置 1

OUT DX, AL

MOV AL, 00001000 ; 对 PC4 置 0

OUT DX, AL

43. 试按照如下要求对 8259A 进行初始化: 系统中只有一片 8259A, 中断请求信号用电平触发方式, 下面要用 ICW4, 中断类型码为 60H、61H、62H.....67H, 用全嵌套方式, 不用缓冲方式, 采用中断自动结束方式。设 8259A 的端口地址为 94H 和 95H。

答: MOV DX,94H ; 偶地址  
MOV AL,00011011B ; ICW1  
OUT DX,AL  
MOV AL,10011111B ; ICW2 , 中断源在 IR7  
MOV DX,95H ; 奇地址  
OUT DX,AL  
MOV AL,00000011B ; ICW4  
OUT DX,AL

44. 试编程对 8253 初始化启动其工作。要求计数器 0 工作于模式 1, 初值为 3000H; 计数器 1 工作于模式 3, 初值为 100H; 计数器 2 工作于模式 4, 初值为 4030H。设端口地址为 40H、41H、42H 和 43H。

答: MOV AL,00011110H ; 控制字  
OUT 43H,AL  
MOV AL,3000H ;计数初值  
OUT 40H,AL  
MOV AL,01010110H ; 计数器 1  
OUT 43H,AL  
MOV AL,100H  
OUT 41H,AL  
MOV AL,10011000H ; 计数器 2  
OUT 43H,AL  
MOV AL,4030H  
OUT 42H,AL

### 模拟试题一

一、简答题:

#### 1.简述 USB 总线的特点。

答: 1)具备即插即用特性, 为 USB 接口设计的驱动程序和应用程序可自动启动、成本低, 节省空间, 为开放性的不具备专利版权的理想工业标准。:

2)可动态连接和重新配置外设, 支持热插拔功能;

3)允许多台设备同时工作;

4)可以向 USB 总线上的设备供电, 总线上的设备可以自备电源;

5)通讯协议支持等时数据传输和异步消息传输的混合模式;

6)支持实时语音、音频、和视频数据传输。

2.什么是中断类型码? 什么叫中断向量? 什么叫中断向量表? 它们之间有什么联系答: 8086/8088 系统可以处理 256 种中断, 为了区别每一种中断, 为每个中断安排一个号码, 称为中断类型码。每一种中断服务程序在内存中的起始地址

称为中断向量，以 32 位逻辑地址表示，即为 CS:IP。把所有中断向量存储在内存中的某一个连续区中，这个连续的存储区称为中断向量表。

中断向量 CS:IP 在中断向量表中的位置为：中断向量表中偏移量为（中断类型码×4）的单元中存放 IP 的值，偏移量为（中断类型码×4+2）的单元中存放 CS 的值。

### 3.简述高速缓冲存储器 Cache 为什么能够实现高速的数据存取？

答：高速缓冲存储器 Cache 是根据程序局部性原理来实现高速的数据存取。即在一个较小的时间间隔内，程序所要使用的指令或数据的地址往往集中在一个局部区域内，因而对局部范围内的存储器地址频繁访问，而对范围外的地址则范围甚少的现象称为程序访问的局部性原理。

如果把正在执行的指令地址附近的一小部分指令或数据，即当前最活跃的程序或数据从主存成批调入 Cache，供 CPU 在一段时间内随时使用，就一定能大大减少 CPU 访问主存的次数，从而加速程序的运行。

### 4.有一个由 20 个字组成的数据区，其起始地址为 3500H：0320H。试写出数据区首末单元的实际地址。

答：数据区首地址 =  $3500\text{H} \times 10\text{H} + 0320\text{H} = 33320\text{H}$

数据区末地址 =  $33320\text{H} + 28\text{H} - 1 = 33347\text{H}$

### 5.设有一个具有 16 位地址和 8 位数据的存储器，问：(1)该存储器能存书多少个字节的信息？(2)如果存储器由 8K×4 位 RAM 芯片组成，需要多少片？(3)需要地址多少位做芯片选择？

答：(1) 因为 8 位二进制数为 1 个字节，所以 16 位地址能存储  $2^{16} = 64\text{KB}$  个字节的信息；

(2) 需要  $64\text{K} \times 8 / 8\text{K} \times 4 = 16$  片 RAM

(3) 因为需要 16 片来构成存储器，而 16 片需要 4 位地址线进行译码输出，故需要 4 位做芯片选择。

### 6.定性分析微型计算机总线的性能指标。

答：微型计算机总线的主要职能是负责计算机各模块间的数据传输，对总线性能的衡量也是围绕这一性能而进行的。性能中最重要的是数据传输率，另外，可操作性、兼容性和性能价格比也是很重要的技术特征。具体来说，总线的主要性能指标有以下几项：

(1) 总线宽度：以位数表示。

(2) 标准传输率 Mb/s：是总线工作频率与总线宽度的字节数之积。

(3) 时钟同步/异步：总线中与时钟同步工作的称为同步总线；与时钟不同步工作的称为异步总线。这取决于数据传输时源模块与目标模块间的协议约定。(4) 信号线数：这是地址总线、数据总线和控制总线线数的总和。信号线数和系统的复杂程度成正比关系。

(5) 负载能力：以系统中可以连接的扩展电路板数表示。

(6) 总线控制方法：包括突发传输、并发工作、自动配置、仲裁方式、逻辑方式、中断方式等项内容。

(7) 扩展板尺寸：这项指标对电路板生产厂家很重要。

(8) 其他指标：电源是 5V 还是 3V，能否扩展 64 位宽度等。

任何系统的研制和外围模块的开发，都必须服从其采用的总线规范。

## 7.虚拟存储器的含义是什么？

答：虚拟存储器是以存储器访问的局部性为基础，建立在主存—辅存物理体系结构上的存储管理技术。在存储系统中，由于主存容量不能满足用户的需要，因而引入辅存作为后援。即辅存做主存用，扩大编程者的使用空间。

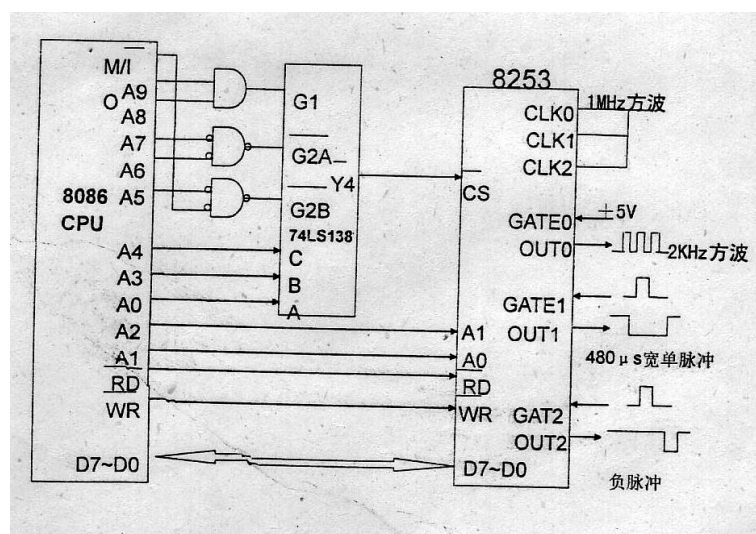
## 二、编程及综合题

1.已知 8255A 的地址为 0060H~0063H，A 组设置方式 1，端口 A 作为输入，PC6PC7 作为输出，B 组设置方式 1，端口 B 作为输入，编制初始化程序。

答案： `MOV DX, 0063H`  
`MOV AL, 00110111B`  
`OUT DX, AL`

2 编写 8253 初始化程序。如下图所示（注意端口地址），要求 3 个计数通道分别完成以下功能：

- (1)通道 0 工作于方式 3，输出频率为 2KHZ 的方波；
  - (2)通道 1 产生宽度为 480us 的单脉冲；
  - (3)通道 2 用硬件方式触发，输出负脉冲，时间常数为 26。
- （提示：8253 的端口地址分别为：0310H、0312H、0314H、0316H）



答案：

编写 8253 的初始化程序

1.确定端口地址：0310H、0312H、0314H、0316H

2.确定工作方式： 通道 0，方式 3  
                  通道 1，方式 1  
                  通道 2，方式 5

3.确定计数值： 通道 0：  $N_0 = 1\text{MHz} / 2\text{KHz} = 500$

通道 1：  $N_1 = 480\mu\text{s} / (1/1\text{mhz}) = 480$

通道 2：  $N_2 = 26$

4. 确定控制字： 通道 0： 00110111B

                  通道 1： 01110011B



---

通道 2: 10011011B

对 3 个通道的初始化程序如下:

; 通道 0 初始化程序

MOV DX, 316H

MOV AL, 00110111B

OUT DX, AL

MOV DX, 310H

MOV AL, 00H

OUT DX, AL

MOV AL, 05H

OUT DX, AL

; 通道 1 的初始化程序

MOV DX, 316H

MOV AL, 001110011B

OUT DX, AL

MOV DX, 312H

MOV AL, 80H

OUT DX, AL

MOV AL, 04H

OUT DX, AL

; 通道 2 初始化程序

MOV DX, 316H

MOV AL, 10011011B

OUT DX, AL

MOV DX, 314H

MOV AL, 26H

OUT DX, AL

## 模式试题二

### 一、 填空:

- 1、设字长为八位, 有  $x = -1$ ,  $y = 124$ , 则有:  $[x+y]_{\text{补}} = 01111011$   $[x-y]_{\text{补}} = 10000011$ ;
- 2、数制转换:  $247.86 =$  \_\_\_\_\_  $H =$  \_\_\_\_\_ BCD;
- 3、在 8086CPU 中, 由于 BIU 和 EU 分开, 所以\_\_\_\_\_和\_\_\_\_\_ 可以重叠操作, 提高了 CPU 的利用率;
- 4、8086 的中断向量表位于内存的\_\_\_\_\_区域, 它可以容纳\_\_\_\_\_个中断向量, 每一个向量占\_\_\_\_\_ 个字节;
- 5、8086 系统中, 地址 FFFF0H 是\_\_\_\_\_ 地址;
- 6、8086CPU 的 MN/MX 引脚的作用是\_\_\_\_\_;
- 7、8251 芯片中设立了\_\_\_\_\_、\_\_\_\_\_ 和\_\_\_\_\_ 三种出错标志;

8、8086CPU 中典型总线周期由\_\_\_\_个时钟周期组成，其中 T1 期间，CPU 输出\_\_\_\_信息；如有必要时，可以在\_\_\_\_两个时钟周期之间插入 1 个或多个 TW 等待周期。

9、8259A 共有\_\_\_\_个可编程的寄存器，它们分别用于接受 CPU 送来的\_\_\_\_命令字和\_\_\_\_命令字。

二、简答题：

1、什么是信号的调制与解调？为什么要进行调制和解调？试举出一种调制的方式。串行长距离通信时，需要利用模拟信道来传输数字信号，由于信道的频带窄，一般为 300~3400HZ，而数字信号的频带相当宽，故传输时必须进行调制，以免发生畸变而导致传输出错。(3 分)

调制是将数字信号→模拟信号。而解调则是相反。例如 FSK 制(调频制或称数字调频)可将数字“1”和“0”分别调制成 2400HZ 和 1200HZ 的正弦波信号。(2 分)

2、已有 AX=E896H，BX=3976H，若执行 ADD BX，AX 指令，则结果 BX，AX，标志位 CF，OF，ZF 各为何值？

BX=220CH (1 分) AX=E896H (1 分) CF=1 (1 分) OF=0 (1 分) ZF=0 (1 分)

三、阅读程序与接口芯片初始化：

1、源程序如下：

```
MOV CL, 4
MOV AX, [2000H]
SHL AL, CL
SHR AX, CL
MOV [2000H], AX
```

试问：① 若程序执行前，数据段内(2000H)=09H，(2001H)=03H，则执行后有(2000H)=\_\_\_\_,(2001H)=\_\_\_\_\_。

② 本程序段的功能\_\_\_\_\_。

2、源程序如下：

```
MOV AL, 0B7H
AND AL, 0DDH
XOR AL, 81H
```

---

```
OR AL, 33H
JP LAB1
JMP LAB2
```

试问：① 执行程序后 AL=\_\_\_\_\_；  
② 程序将转到哪一个地址执行：\_\_\_\_\_。

3、源程序如下：

```
MOV CX, 9
MOV AL, 01H
MOV SI, 1000H
NEXT: MOV [SI], AL
      INC SI
      SHL AL, 1
      LOOP NEXT
```

试问：① 执行本程序后有：AL=\_\_\_\_\_；SI=\_\_\_\_\_；CX=\_\_\_\_\_；  
② 本程序的功能是\_\_\_\_\_。

4、某系统中 8253 占用地址为 100H~103H。初始化程序如下：

```
MOV DX, 103H
MOV AL, 16H
OUT DX, AL
SUB DX, 3
OUT DX, AL
```

试问：① 此段程序是给 8253 的哪一个计数器初始化？安排工作在何种工作方式？\_\_\_\_\_；

② 若该计数器的输入脉冲的频率为 1MHZ，则其输出脉冲的频率为：\_\_\_\_\_。

5、已知某 8255A 在系统中占用 88~8BH 号端口地址，现欲安排其 PA，PB，PC 口全部为输出，PA，PB 口均工作于方式 0 模式，并将 PC<sub>6</sub>置位，使 PC<sub>3</sub>复位，试编写出相应的初始化程序：

#### 模式试题二 参考答案

##### 一、填空题

2、F7.DCH      001001000111.10000110 BCD

3、取指令      执行指令

4、00000H~003FFH 区      256 个      4 个

5、CPU 复位以后执行第一条指令的地址

6、决定 CPU 工作在什么模式(最小/最大)

7、奇/偶错      帧格式错      溢出错

8、4 个      地址      T<sub>3</sub> 和 T<sub>4</sub>



## 9、7个 初始化 操作

三、阅读程序与接口芯片初始化：

1、 (2000H)=39H      (2001H)=00H

将(2000H),(2001H)两相邻单元中存放的未组合型 BCD 码压缩成组合型 BCD 码，  
并存入(2000H)单元，0→(2001H)

2、 37H      LAB<sub>2</sub>

3、 0      1009H      0

对数据段内 1000H~1008H 单元置数，依次送入 1, 2, 4, 8, 16, 32, 64, 128,  
0 共九个

4、计数器 0      工作于方式 3      45.454KHZ

5、      MOV AL, 80H

        OUT 8BH, AL

        MOV AL, 0DH

        OUT 8BH, AL

        MOV AL, 06H

        OUT 8BH, AL

模式试题三

### 一、填空题

1、将十进制数 279.85 转换成十六进制数、八进制数、二进制数及 BCD 码数分别为：\_\_\_\_\_H, \_\_\_\_\_Q, \_\_\_\_\_B, \_\_\_\_\_BCD。

2、字长为 8 位的二进制数 10010100B，若它表示无符号数，或原码数，或补码数，则该数的真值应分别为\_\_\_\_\_D, \_\_\_\_\_D 或 \_\_\_\_\_D。

3、已知 BX=7830H, CF=1, 执行指令: ADC BX, 87CFH 之后, BX=\_\_\_\_\_, 标志位的状态分别为 CF=\_\_\_\_\_, ZF=\_\_\_\_\_, OF=\_\_\_\_\_, SF=\_\_\_\_\_。

4、8086 中，BIU 部件完成\_\_\_\_\_功能，EU 部件完成 \_\_\_\_\_功能。

5、8086 中引脚  $\overline{\text{BHE}}$  信号有效的含义表示\_\_\_\_\_。

6、8086 正常的存储器读/写总线周期由\_\_\_\_\_个 T 状态组成，ALE 信号在 \_\_\_\_\_状态内有效，其作用是\_\_\_\_\_。

7、设 8086 系统中采用单片 8259A，其 8259A 的 ICW<sub>2</sub>=32H，则对应 IR<sub>5</sub> 的中断类型为\_\_\_\_\_H，它的中断入口地址在中断向量表中的地址为 \_\_\_\_\_H。

### 二、简答及判断题

1、某指令对应当前段寄存器  $CS=FFFFH$ ，指令指针寄存器  $IP=FF00H$ ，此时，该指令的物理地址为多少？指向这一物理地址的  $CS$  值和  $IP$  值是唯一的吗？试举例说明

2、8086CPU 的  $FLAG$  寄存器中，状态标志和控制标志有何不同？程序中是怎样利用这两类标志的？

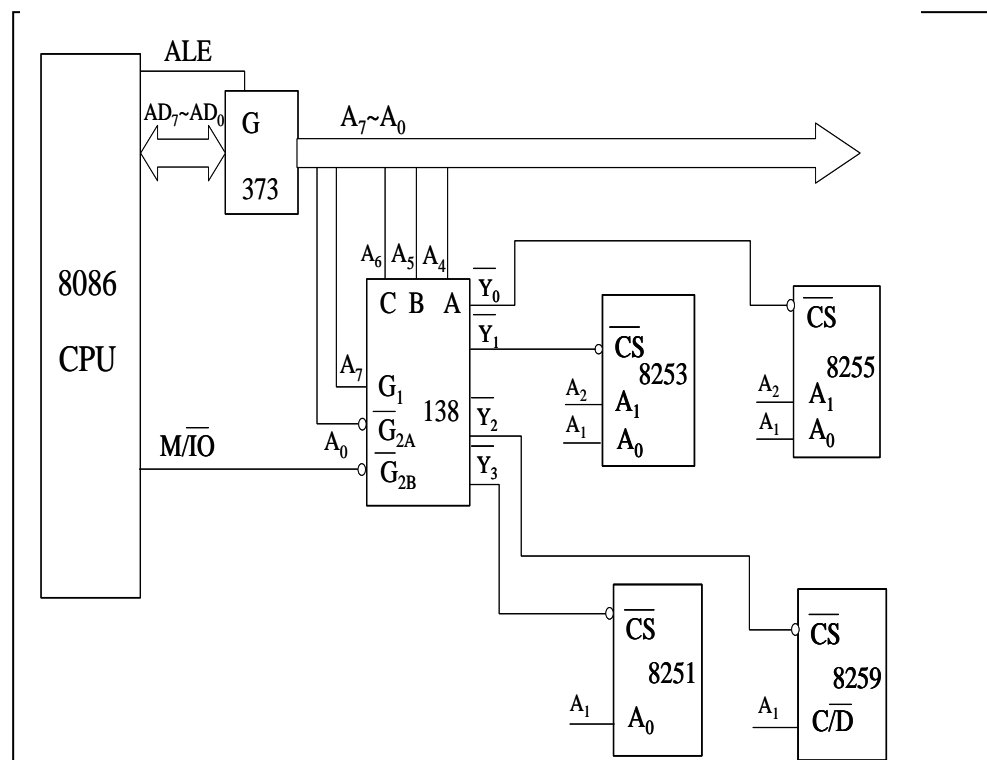
3、设采用 16550 进行串行异步传输，每帧信息对应 1 个起始位，7 个数据位，1 个奇/偶校验位，1 个停止位，波特率为 4800，则每分钟能传输的最大字符数为多少个？

### 三、读图和作图题

1、8086 系统中接口连接关系如下图所示。要求回答以下问题：

(1) 试分别确定 8255，8253，8259 及 8251 的端口地址；

(2) 设 8255 的  $PA$  口为输出， $PB$  口为输入，试写出对  $PA$  口和  $PB$  口执行输入/输出操作的指令。



8255 的端口地址为：\_\_\_\_\_；

8253 的端口地址为：\_\_\_\_\_；

8259 的端口地址为：\_\_\_\_\_；

8251 的端口地址为：\_\_\_\_\_；

对  $PA$  口操作的 I/O 指令为\_\_\_\_\_；

对  $PB$  口操作的 I/O 指令为\_\_\_\_\_。

## 2、作图题。

系统采用 4 个接口芯片：8253，8251，8259 及 8255。要求 8253 的通道 0 用作实时时钟，每当定时时间到之后向 8259 的  $IR_2$  送入中断申请信号。8253 通道 1 用作方波发生器作为 8251 的收发时钟脉冲。8253 通道 0，通道 1 的门控信号由 8255 PC 口的  $PC_3$  和  $PC_2$  控制。

(1) 画出 4 个芯片之间控制线的连接图；

(2) 8253 的两个通道应分别工作在什么方式？

## 四、程序阅读题

### 1、源程序如下：

```
MOV AH, 0
MOV AL, 9
MOV BL, 8
ADD AL, BL
AAA
AAD
DIV AL
```

结果 AL\_\_\_\_\_, AH=\_\_\_\_\_,BL=\_\_\_\_\_。

### 2、源程序如下：

```
MOV AX, SEG TABLE ; TABLE 为表头
MOV ES, AX
MOV DI, OFFSET TABLE
MOV AL, '0'
MOV CX, 100
CLD
REPNE SCASB
```

问：1) 该段程序完成什么功能？

2) 该段程序执行完毕之后，ZF 和 CX 有几种可能的数值？各代表什么含义？

### 3、源程序如下：

```
CMP AX, BX
JNC L1
JZ L2
JNS L3
JNO L4
JMP L5
```

设  $AX=74C3H$ ,  $BX=95C3H$ , 则程序最后将转到哪个标号处执行？试说明理由。

### 4、源程序如下：

```

MOV DX, 143H
MOV AL, 77H
OUT DX, AL
MOV AX, 0
DEC DX
DEC DX
OUT DX, AL
MOV AL, AH
OUT DX, AL

```

设 8253 的端口地址为 140H~143H，问：

(1)程序是对 8253 的哪个通道进行初始化？

(2)该通道的计数常数为多少？

(3)若该通道时钟脉冲 CLK 的周期为 1μs，则输出脉冲 OUT 的周期为多少 μs？

## 五、编程题

1、8255 的编程。设 8255 的端口地址为 200H~203H。

(1)要求 PA 口方式 1，输入；PB 口方式 0 输出；PC7~PC6 为输入；PC1~PC0 为输出。试写出 8255 的初始化程序。

(2)程序要求当 PC7=0 时置位 PC1，而当 PC6=1 时复位 PC0，试编制相应的程序。2、自 BUFFER 开始的缓冲区有 6 个字节型的无符号数：10，0，20，15，38，236，试编制 8086 汇编语言程序，要求找出它们的最大值、最小值及平均值，分别送到 MAX、MIN 和 AVI 三个字节型的内存单元。要求按完整的汇编语言格式编写源程序。

## 模式试题三参考答案

### 一、填空题

1、117.D99H    427.6631Q    000100010111.110110011001B

0010 01111001.1000 0101 BCD

2、148D    -20D    -108D    3、BX=0000H    CF=1    ZF=1    OF=0    SF=0

4、总线接口功能    指令的译码及执行功能    5、高 8 位数据线 D15~D8 有效

6、4 T1 给外部的地址锁存器提供一个地址锁存信号    7、35H

000D4H~000D7H

### 二、简答及判断题

1、∴    FFFF0

```

      +   FF00
-----
 1  0FEF
  |
  |

```

自然丢失

故物理地址为 0FEF0H。指向该物理地址的 CS, IP 值不唯一。

例如: CS: IP=0000:FEF0H 也指向该物理地址。

2、状态标志表示算术运算或逻辑运算执行之后,运算结果的状态,这种状态将作为一种条件,影响后面的操作。控制标志是人为设置的,指令系统中有专门的指令用于控制标志的设置或清除,每个控制标志都对某一特定的功能起控制作用。

3、每帧占 1+7+1+1=10 位,波特率为 4800 bit/s,故每分钟能传送的最大字符数为 28800(个)

$$\frac{4800 \times 60}{10} = 28800 \text{ 个}$$

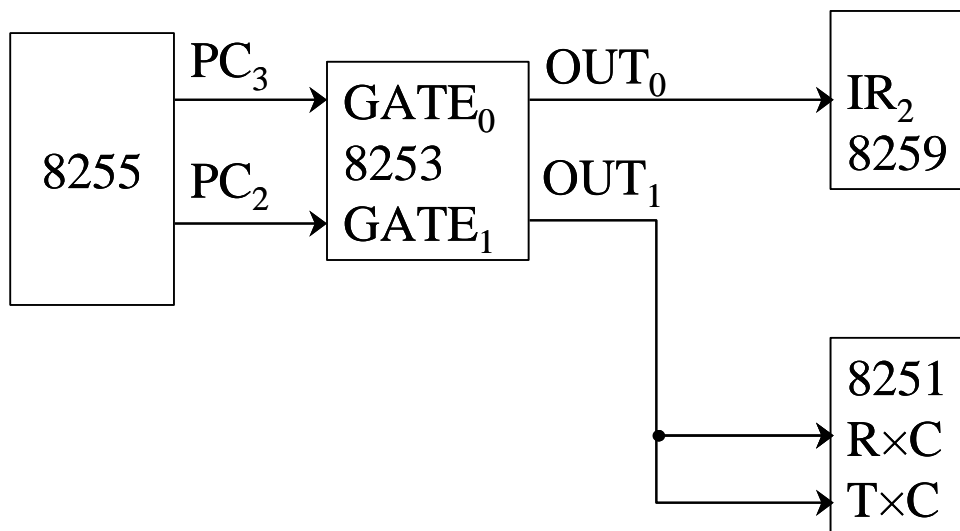
### 三、读图和作图题

1、	(1)	A <sub>7</sub>	A <sub>6</sub>	A <sub>5</sub>	A <sub>4</sub>	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	
		1	0	0	0	任意	×	×	0	Y <sub>0</sub> ——
		1	0	0	1	任意	×	×	0	Y <sub>1</sub> ——
		1	0	1	0	任意	×	×	0	Y <sub>2</sub> ——
		1	0	1	1	任意	×	×	0	Y <sub>3</sub> ——

∴ 8255 的端口地址为 80H, 82H, 84H, 86H  
8253 的端口地址为 90H, 92H, 94H, 96H  
8259 的端口地址为 A0H, A2H,  
8251 的端口地址为 B0H, B2H,

(2) OUT 80H, AL          IN    AL,    82H

2、 (1) 控制线连接图如图所示。



### 四、程序阅读题

1、AL=01H    AH=00H    BL=08H

2、(1) 从目的串中查找是否包含字符 '0', 若找到则停止, 否则继续重复搜索。

**ZF=0, 说明未找到字符**

**CX=0, 且 ZF=0 说明串中无字符 '0'**

– 95C3H

DF00H

∴ 程序将转到 L<sub>5</sub> 标号处执行。

(2)计数常数为 10000D, BCD 计数。

(3)工作在方式 3，方波速率发生器      周期=10000×1μs=10000μS=10ms

1、(1) MOV DX, 203H      MOV AL, 10111000B      OUT DX, AL  
(2)

**TEST AL, 80H**

**JNZ** **NEXT1****MOV DX, 203H**

**MOV AL, 00000011B ; 对 PC1 置位**

**OUT DX, AL (2 分)**

**NEXT1: MOV AL, AH**

**TEST AL, 40H**

**.JZ      NEXT2**

**MOV AL, 00000000B ; 对 PC0 复位**

**MOV DX, 203H**

**OUT DX, AL**

**NEXT2:** ..... (3 分)

## 2. DATA SEGMENT

**BUFER DB 10, 0, 20, 15, 38, 236**

**MAX DB 0**

**MIN DB 0****AVI**      **DB**    **0**

**DATA      ENDS      (2 分)**

[illegible]

**DW      100    DUP (?)**

## STACK ENDS

CODE	SEGMENT
00000000	00000000
00000001	00000001
00000002	00000002
00000003	00000003
00000004	00000004
00000005	00000005
00000006	00000006
00000007	00000007
00000008	00000008
00000009	00000009
0000000A	0000000A
0000000B	0000000B
0000000C	0000000C
0000000D	0000000D
0000000E	0000000E
0000000F	0000000F
00000010	00000010
00000011	00000011
00000012	00000012
00000013	00000013
00000014	00000014
00000015	00000015
00000016	00000016
00000017	00000017
00000018	00000018
00000019	00000019
0000001A	0000001A
0000001B	0000001B
0000001C	0000001C
0000001D	0000001D
0000001E	0000001E
0000001F	0000001F
00000020	00000020
00000021	00000021
00000022	00000022
00000023	00000023
00000024	00000024
00000025	00000025
00000026	00000026
00000027	00000027
00000028	00000028
00000029	00000029
0000002A	0000002A
0000002B	0000002B
0000002C	0000002C
0000002D	0000002D
0000002E	0000002E
0000002F	0000002F
00000030	00000030
00000031	00000031
00000032	00000032
00000033	00000033
00000034	00000034
00000035	00000035
00000036	00000036
00000037	00000037
00000038	00000038
00000039	00000039
0000003A	0000003A
0000003B	0000003B
0000003C	0000003C
0000003D	0000003D
0000003E	0000003E
0000003F	0000003F
00000040	00000040
00000041	00000041
00000042	00000042
00000043	00000043
00000044	00000044
00000045	00000045
00000046	00000046
00000047	00000047
00000048	00000048
00000049	00000049
0000004A	0000004A
0000004B	0000004B
0000004C	0000004C
0000004D	0000004D
0000004E	0000004E
0000004F	0000004F
00000050	00000050
00000051	00000051
00000052	00000052
00000053	00000053
00000054	00000054
00000055	00000055
00000056	00000056
00000057	00000057
00000058	00000058
00000059	00000059
0000005A	0000005A
0000005B	0000005B
0000005C	0000005C
0000005D	0000005D
0000005E	0000005E
0000005F	0000005F
00000060	00000060
00000061	00000061
00000062	00000062
00000063	00000063
00000064	00000064
00000065	00000065
00000066	00000066
00000067	00000067
00000068	00000068
00000069	00000069
0000006A	0000006A
0000006B	0000006B
0000006C	0000006C
0000006D	0000006D
0000006E	0000006E
0000006F	0000006F
00000070	00000070
00000071	00000071
00000072	00000072
00000073	00000073

**ASSUME CS: CODE, DS: DATA, SS: STACK (1分)**

**START      PROC      FAR**

**BEGIN:    PUSH       DS**

MOV AX, 0

**PUSH      AX**

MOV AX, DATA

---

```

        MOV    DS, AX
        LEA    DI, BUFFER
        MOV    DX, 0    ; 使 DH=0, DL=0
        MOV    CX, 6
        MOV    AX, 0    ; 和清 0
        MOV    BH, 0    ; 最大值
        MOV    BL, 0FFH ; 最小值                (2 分)
LOP1:   CMP    BH, [DI]
        JA     NEXT1    ; 若高于转移
        MOV    BH, [DI]; 大值→BH
NEXT1:  CMP    BL, [DI];
        JB     NEXT2    ; 若低于转移
        MOV    BL, [DI]; 小值→BL                (2 分)
NEXT2:  MOV    DL, [DI]; 取一字节数据
        ADD    AX, DX; 累加和
        INC    DI
        LOOP   LOP1
        MOV    MAX, BH; 送大值
        MOV    MIN, BL; 送小值                (3 分)
        MOV    DL, 6
        DIV    DL,      ; 求平均值
        MOV    AVI, AL; 送平均值
        RET
START   ENDP
CODE    ENDS
        END    BEGIN                (3 分)

```

