第一章 微型计算机基础

1. ：概述

一 微型计算机发展

微型计算机:CPU做在一个集成电路芯片上。

第一代微型处理器和微型计算机 1971年

属于4位和低档8位的微处理器时代

典型产品：微处理器：Intel 4004 Intel 8008

微型计算机：MCS—4 MCS—8

集成度：2000只晶体管/芯片 平均指令执行时间：20μs

第二代微型处理器和微型计算机 1974年

属于成熟8位的微处理器时代

典型产品：微处理器：Intel 8080 MC6800 Z—80

微型计算机：APPLE—I

集成度：5千~1万只晶体管/芯片 平均指令执行时间：1~2μs

第三代微型处理器和微型计算机 1978年

属于16位的微处理器时代

典型产品： Intel 8088 Intel 8086 Intel 802886

IBM—PC/XT IBM—PC/AT

集成度：2~13.4万只晶体管/芯片 平均指令执行时间：0.5μs

第四代微型处理器和微型计算机 1978年

属于32位的微处理器时代

典型产品： Intel 80386（27.5万） Intel 80486（120万） MC68020

IBM—PC/386 IBM—PC/486

集成度：27~120万只晶体管/芯片 平均指令执行时间：0.1μs

技术特点：①分段和分页存储管理

②多级流水CPU

③CPU内含8KB cache（Intel 80486）

第五代微型处理器和微型计算机 1993年

64位数据通路

1.Pentium 1993年 （相当于80586）

集成度：310万只晶体管/芯片

技术特点：①超标量流水线（二条流水线）

②CPU内含指令cache和数据cache

③动态转移预测

2.Pentium Pro 1995年

集成度：550万只晶体管/芯片

技术特点：①CPU内含三条流水线 14级流水

②Cisc/Risc混合使用

③分支预测

3.Pentium MMX 1996年

增加57条多媒体指令

4.Pentium II 1997年

集成度：750万只晶体管/芯片

①双重独立总线结构

②工作电压 2.8V 工艺0.35μm

5.Pentium III 1999年

集成度：950万只晶体管/芯片~ 2800万只晶体管/芯片（后期）

①工作电压 2.8V 最低 1.8V 工艺0.18μm（后期）

6.Pentium IV 2000~2007年

集成度：4200万只晶体管/芯片~ 1.78亿只晶体管/芯片

①工作电压 1.2V 工艺0.13μm 采用NetBurst技术

第六代微型处理器和微型计算机 2007~至今

采用Core架构（多核） 工作电压 0.85V 工艺45nm

Core 2（DVD）（双核） Core 2（Quad）（四核）

2.91亿 5.82亿

每核4条流水线 每核4条流水线

14级流水 14级流水

共4×2=8条 共4×4=16条流水线

4×4×14=224条指令

1. ：微型机的分类和应用

1.按字长：4、8、16、32位

2.按结构：单片机 单板机 多板机

1. ：微型计算机基础知识

一：微型计算机的硬件结构

冯·诺依曼设计思想

1. 五大部件：运算器 控制器 存储器 输入部件 输出部件

（运算器和控制器合为中央处理部件（CPU））

1. 存储程序
2. 二进制

哈佛结构：指令存储器和数据存储器

二：微型计算机软件

系统软件： DOS→Windows

第二章 微处理器（CPU）

由于半导体制造工艺影响，微处理器的结构受到下面几方面影响：

1. 引脚数的限制
2. 芯片面积限制
3. 器件速度限制

为了解决上述问题，微处理器结构采取下列措施：

1. 引脚功能复用
2. 总线 累加器结构
3. 可控三态电路
4. 总线分时复用
5. ：8086/8088的技术特点

⑴16位内部体系结构

⑵指令系统基本完备

⑶多种寻址方式

⑷20位地址线 寻址范围：1M字节存储空间

⑸16位I/O端口线，可寻址64K端口地址

⑹有较强中断功能

⑺具有DMA功能

⑻多处理器功能

8086内部数据总线16位，外部数据总线16位

8088内部数据总线16位，外部数据总线8位

1. ：8086/8088存储器组织和I/O组织

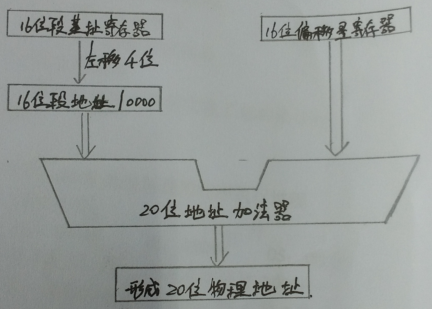
一：8086/8088存储器组织

1. 逻辑地址和物理地址

物理地址：实际地址（20位物理地址）

逻辑地址：在程序中使用，由段基址和偏移量组成

2.8086存储器20位物理地址的形成8086存储器20位物理地址的形成

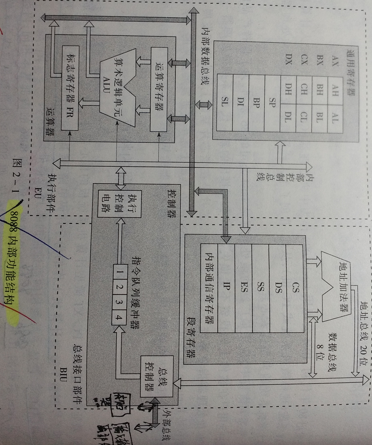


二：8086的I/O组织

1. I/O端口地址：独立编址（有64K端口地址）
2. 访问I/O端口，用In/Out指令

访问存储器，用mov指令

1. ：8086的编程结构



一：总线接口部件（BIU）

功能：负责与存储器、I/O接口传送数据

组成：1. 四个段寄存器

16位 CS—代码段寄存器

16位 DS—数据段寄存器

16位 ES—附加数据段寄存器

16位 SS—堆栈段寄存器

2. 指令指针寄存器（IP）

CS·IP↔程序计数器PC （逻辑地址，CS段基址，IP偏移量）

指令的20位物理地址=CS左移4位＋IP

3.地址加法器

4.指令队列

二：执行部件（EU）

功能：负责指令的执行

组成：1. 通用寄存器

|  |  |
| --- | --- |
| 8位 AH | 8位 AL |

16位AX

AXAL 累加器

|  |  |
| --- | --- |
| 8位 BH | 8位 BL |

16位BX

|  |  |
| --- | --- |
| 8位 CH | 8位 CL |

16位CX

|  |  |
| --- | --- |
| 8位 DH | 8位 DL |

16位DX

16位 SP、BP、DI、SL

2. ALU

3. 标志寄存器（FR）

ZF：结果为0标志位 ZF=1结果为0，ZF=0结果不为0

CF：进位标志 最高位产生进位，CF=1；否则CF=0

OF：溢出标志 溢出时，OF=1；否则OF=0

SF：符号标志 结果为负，SF=1；否则SF=0

PF：奇偶标志 结果中“1”的个数为偶数，PF=1；否则PF=0

IF：中断允许标志 IF=1允许CPU响应可屏蔽中断；IF=0禁止CPU接受外界的可屏

蔽中断

TF：单步跟踪标志

DF：方向标志

AF：辅助进位标志

三：流水线结构

通过使用指令队列实现了总线接口部件BIU和执行部件EU并行流水操作，提高了CPU效率。

1. ：8086微处理器的程序执行过程

一：系统启动

CS：FFFFH IP：0000H DS：0000H ES：0000H 下一条指令地址

指令队列：空 标志寄存器 0000H CS·IP↔程序计数器PC

1. 指令的物理地址：CS左移4位＋IP

逻辑地址：CS IP

基本输入输出系统BIUS（ROM）

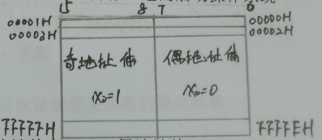
功能：自检和初始化硬件系统

把操作系统的核心部分从外存调入内存，进而启动操作系统

含有若干基本驱动程序

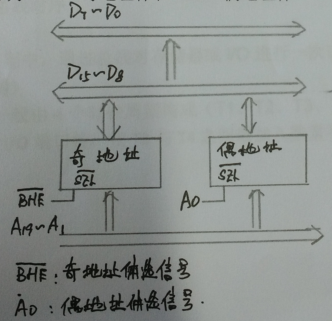
1. ：8086存储体结构及读写

1.8086存储体的结构



①把1MB存储体分为512KB奇地址体和512KB偶地址体

②交叉编址

****

读写：

从偶地址存取一个字节 1 0 D7—D0

从奇地址存取一个字节 0 1 D15—D8

从偶地址存取一个字（规则字 16位） 0 0 D15—D0

从奇地址存取一个字（非规则字 16位） 0 1 D15—D8

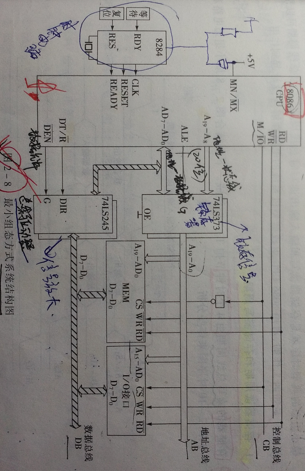
1 0 D7—D0

1. ：8086工作模式

最小模式：系统中只有一个8086微处理器，所有总线控制信号由8086发出。

最大模式：系统中有两个或以上的微处理器，所有总线控制信号由总线控制器发出。

一：最小模式



①MN/MX：最大最小模式选择 （接+5V 最小模式，接地 最大模式）

②AD15—AD0：地址/数据复用线（低16位）

③A19—A16：高4位地址

④RD：读 WR：写

⑤M/IO：内存/外设选择 高电平，内存：低电平，外设

⑥READY：准备好为1。数据已有效在数据总线上。

⑦ALE：锁存信号 CLK：时钟信号 Reset：复位

DEN：74LS245总线驱动器的使能信号

DT/R：74LS245总线驱动器的方向控制

INTR：可屏蔽中断请求（外设→CPU）

INTA：中断响应（CPU→外设）

NMI：不可屏蔽中断请求

二：最大模式

1.总线控制器：解决多个处理器之间协调工作，对总线共享问题。

S0，S1，S2 总线周期状态信号

DEN DT/R M/IO

0 0 0 → INTA

0 0 1 → IORC

0 1 0 → IOWC AIOWC

0 1 1 → 暂停

1 0 0 → MRDC

1 0 1 → MRDC

1 1 0 → MWTC AMWTC

1 1 1 → 无源

2.总线仲裁器 8289

按照一定的规程，把总线使用权赋给优先级高的微处理器

1. ：8086的总线时序

指令周期：

总线周期：指令周期中，通过总线对存储器或I/O进行一次读/写所花的时间

时钟周期：（T周期）

8086总线周期：一般由4个时钟周期构成（T1、T2、T3、T4）

如果存储器/ I/O端口慢，在T3与T4之间需加入数量不等的等待周期Tw

一：读周期

T1状态：

①送地址（20位）

②M / I/o 有效，选择内存/ I/o 端口

③送BHE信号

④锁存信号ALE有效

T2状态：读信号有效

T3状态：若内存或外设足够快，CPU接收数据

若内存或外设较慢，则需在T3之后加入Tw，直到Ready为高

T4状态：CPU对数据总线采样，获得数据

第三章 微型计算机的指令系统

1. ：寻址方式

指令的寻址：⑴顺序方式

⑵跳转方式

操作数寻址：

1. 立即寻址：操作数包含在指令中，操作数紧跟操作码一起存放代码段

例： MOV AL， 05H → AL=05H

1. 寄存器寻址：操作数包含在CPU的寄存器中

例： MOV AL， BL

1. 直接寻址：操作数的偏移地址包含在指令中，紧跟操作码存放于代码段，操作数存放于数据段。 [ ] 表示地址

例： MOV AX，DS [2000H] = MOV AX， [2000H]

设DS=1000H [12000H]=1111H

DS左移4位＋2000H=10000H+2000H=12000H

（AX）=1111H

1. 寄存器间接寻址：操作数的偏移地址在寄存器中，操作数在存储器中
2. 如果此寄存器为BX、SI、DI，则操作数在数据段中，操作数的物理地址：DS左移4位+BX或SI或DI
3. 如果此寄存器为BP，则操作数在堆栈中，操作数的物理地址：SS左移4位+BP

例：MOV AX，[BX] 设[AX]=0A364H （BX）=1010H

（DS）=3000H （31010H）=3445H

解：操作数物理地址=DS左移4位+BX=30000H+1010H=31010H

则： （AX）=3445H

1. 变址寻址：指定的寄存器中内容+指令中给出的8位或16位偏移量作为操作数地址
2. 操作数物理地址=DS左移4位+BX或SI或DI+8位或16位偏移量
3. 操作数物理地址=SS左移4位+BP+8位或16位偏移量

例：MOV AX，0010H[BX] 设 （DS）=3000H （BX）=0100H

（30110H）=6133H

解：操作数物理地址=DS左移4位+BX+偏移量

=30000H+0100H+0010H=30110H

则： （AX）=6133H

1. 基址加变址寻址

操作数的偏移地址（有效地址）=基址寄存器（BX或BP）+变址寄存器（SI或DI）+8位或16位偏移量

1. 操作数物理地址=DS左移4位+BX+SI或DI+8位或16位偏移量
2. 操作数物理地址=DS左移4位+BP+SI或DI+8位或16位偏移量

例：MOV AX，0010H[BX][DI] 设 （DS）=4000H （BX）=0100H

（DI）=0001H （40111H）=6789H

解：操作数物理地址=DS左移4位+BX+DI+偏移量

=40000H+0100H+0001H+0010H=40111H

则： （AX）=6789H

1. ：指令系统

一：数据传送指令

1. MOV指令

格式：MOV DST， SRC

操作：（DST）←（SRC） DST：目的操作数 SRC：源操作数

MOV指令的六个约束规定

①目的操作数不能为常数 例：MOV 05H，AL（×）

②指令中的源和目的操作数不能同时为存储单元 例：MOV [2000H]，[1000H]（×）

③不能直接将常数赋给段寄存器 例：MOV DS，3000H（×）

MOV AX，3000H

（√）

MOV DS，AX

④不能将任何数赋给CS代码寄存器 例：MOV CS，AX（×）

⑤段寄存器之间不能传送数据（CS、DS、ES、SS）

例：MOV DS，ES（×） MOV AX，ES

（√）

MOV DS，AX

⑥不同类型的数据不能互相传送

例：MOV AL，BX（×） MOV BL，3064H（×）

MOV AX，BX（√） MOV BX，3064H（√）

1. 堆栈操作指令

①进栈指令PUSH

格式：PUSH SRC

功能：（SP）←（SP）－2，把SRC中的16位操作数压入堆栈

②出栈指令POP

格式：POP DST

功能：从栈顶弹出16位操作数到目的操作数中

（SP）←（SP）＋2

注意：目的操作数不能为常数

PUSH 3064H（×） MOV AX，3064H

（√）

PUSH AX

1. 交换指令XCHG

格式：XCHG （OPR1），（OPR2）

操作：（OPR1）↔（OPR2）

说明：有一个操作数必须是在通用寄存器中，另一个操作数可以是通用寄存器或存储单元，

不能使用段寄存器（CS、DS、ES、SS）和立即数。

例：（DS）=2000H，（AX）=FEDCH，（SI）=0045H，（20046H）=5454H

执行指令：XCHG AX，[SI+01H]

解：操作数物理地址=DS左移4位+SI+01H

=20000H+0045H+01H=20046H

又（20046H）=5454H

则：（AX）=5454H （20046H）=FEDCH

2：累加器专用传送指令

1. IN输入指令 外设→CPU

①长格式：I/O端口地址 ≤ FFH

IN AL或AX ，外设端口地址

②短格式：I/O端口地址 > FFH

MOV DX，外设端口地址

IN AL或AX，DX

例：用指令把外设端口6A3CH的字节内容送BYTE存储单元中

解： MOV DX，6A3CH 外设→CPU

MOV AL， DX

MOV BYTE，AL CPU→内存

1. OUT输出指令 CPU→外设

①长格式：外设端口地址 ≤ FFH

OUT 外设端口地址，AL或AX

②短格式：外设端口地址 > FFH

MOV AL或AX，外设端口地址

OUT DX，AL或AX

例：把上述中的BYTE存储单元的内容送到外设10H

解： MOV AL，BYTE， 内存→CPU

OUT 10H，AL， CPU→外设

1. 地址传送指令
2. LEA 有效地址送寄存器

格式：LEA REG，SRC

功能：把源操作数SRC有效地址（偏移地址）送寄存器REG

1. LDS 指针送寄存器和DS

格式：LDS REG，SRC

功能：把源操作数SRC指定的连续4个存储单元的前两个存储单元内容送REG寄存器，后两个存储单元内容送DS中。

例：设（SI）=0002H，（BX）=0020H， 执行指令：LEA DI，[BX+10H]

解：[BX+10H]偏移地址=BX+10H=0030H

（DI）=0030H

例：设当前状态（SI）=0002H，（DS）=0A00H，（BX）=0010H，（0A010H）=0F0D9H

（0A012H）=3C4DH，执行指令 LDS SI，[BX]↔LDS SI，DS:[BX]

|  |
| --- |
|  |
| D9H |
| F0H |
| 4DH |
| 3CH |

解：物理地址：DS左移4位+BX=0A000H+0010H=0A010H

（SI）=0F0D9H （DS）=3C4DH

4.标志寄存器传送指令

1. 标志寄存器进栈指令

格式：PUSHF

1. 标志寄存器出栈指令

格式：POPF

二：算术运算指令

1. 加法指令
2. 一般加指令 ADD

格式：ADD 目的操作数，源操作数

功能：源操作数＋目的操作数，结果放在目的操作数，并根据结果设置标志位。

1. 带进位加法指令 ADC

格式：ADC 目的操作数，源操作数

功能：源操作数＋目的操作数＋CF，结果放在目的操作数

1. 加1指令 INC

格式：INC 目的操作数

功能：目的操作数＋1→目的操作数

1. 减法指令
2. 一般减指令 SUB

格式：SUB 目的操作数，源操作数

功能：目的操作数－源操作数→目的操作数

1. 借位减指令 SBB

格式：SBB 目的操作数，源操作数

1. 减1指令 DEC

格式：DEC 目的操作数

功能：目的操作数－1→目的操作数

1. 求负（求补）指令 NEG

格式：NEG 目的操作数

功能：0－目的操作数→目的操作数

1. 比较指令 CMP

格式：CMP 操作数1，操作数2

功能：操作数1－操作数2，并不保存结果，但根据结果设置标志位。

1. 乘法指令
2. 无符号乘法指令 MUL

格式：MUL SRC

功能：把由SRC指定的无符号数与AL或AX的内容相乘。

①两个字节相乘→AX （16位） （SRC）×（AL）→（AX）

②两个字相乘→DX：AX （高16，低16）（SRC）×（AX）→（DX：AX）（32位）

1. 有符号乘法指令 IMUL
2. 除法指令
3. 字节转为字指令 CBW

格式：CBW

功能：把AL寄存器内容的符号扩展到AX寄存器

说明：①如果AL中数据的最高有效值为0，则（AH）=00H

②如果AL中数据的最高有效值为1，则（AH）=0FFH

1. 字转为双字指令 CWD

格式：CWD

说明：①如果AX中数据的最高有效值为0，则（DX）=0000H

②如果AX中数据的最高有效值为1，则（DX）=0FFFFH

1. 无符号数除法指令 DIV

格式：DIV SRC

功能：把AX或DX：AX的内容除以SRC指定的无符号数

①16位除以8位，商（8位）存放于AL中，余（8位）存放于AH中

②32位除以16位，商（16位）存放于AX中，余（16位）存放于DX中

例：写出实现 34H×65H的指令

MOV AL，34H

MOV BL，65H

MUL BL （AL）×（BL）→（AX）

例：写出实现 2056H÷ 36H的指令

MOV AX，2056H

MOV BL，36H

MUL BL （AX）÷（BL）→（AL）……（AH）

三：逻辑指令

1. 逻辑与指令 AND

格式：AND 目的操作数，源操作数

功能：目的操作数和源操作数按位相与，结果送目的操作数

例：设（BL）=0AFH 执行指令：AND BL，7FH

1 0 1 0 1 1 1 1

0 1 1 1 1 1 1 1

0 0 1 0 1 1 1 1 （BL）=00101111B=2FH

1. 逻辑或指令 OR

格式：OR 目的操作数，源操作数

功能：目的操作数和源操作数按位相或，结果送目的操作数

1. 逻辑非指令 NOT

格式：NOT 目的操作数

功能：目的操作数按位取反，结果送目的操作数

1. 逻辑异或指令 XOR

格式：XOR 目的操作数，源操作数

功能：目的操作数和源操作数按位相异或，结果送目的操作数

四：测试指令 TEST

格式：TEST 目的操作数，源操作数

功能：目的操作数和源操作数按位相与，并不保存结果，根据结果设置标志位。

例：若要检测AL中的最低位是否为1，为1则转移。

TEST AL，01H

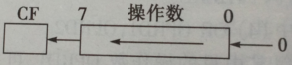
JNZ THERE（为1转移）

五：移位指令

1. 逻辑左移 SHL

格式：SHL 操作数，（移位次数）COUNT

功能：对给定的操作数，按位左移1位或CL位，每次移位时，最高位移入CF中，最低位补零。



说明：①操作数不能为常数

②如果移位次数大于1，则移位次数需写入CL寄存器中

例：设（AL）=01H

SHL AL，1 （AL）=0000 0010=02H

SHL AL，2 （×） MOV CL，2

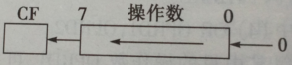
（√）

SHL AL，CL

1. 算术左移指令 SAL

格式：SAL 操作数，（移位次数）COUNT

功能：同SHL



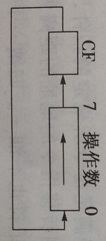
1. 循环右移指令 ROL

格式：ROL 操作数，（移位次数）COUNT



1. 带进位循环左移指令 RCL

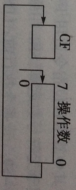
格式：RCL 操作数，（移位次数）COUNT



1. 逻辑右移指令 SHR

格式：SHR 操作数，（移位次数）COUNT

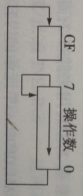
功能：对给定的操作数，右移COUNT次，最低位移入CF中，最高位补零。



1. 算术右移指令 SAR

格式：SAR 操作数，（移位次数）COUNT

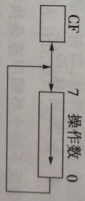
功能：与逻辑右移SHR不同处，即最高位不变。



1. 循环右移指令 ROR

格式：ROR 操作数，（移位次数）COUNT

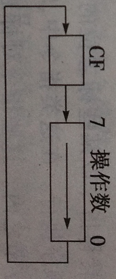
功能：对给定的操作数，右移COUNT次，每次移位时，最低位移入CF中，并把该值移入空处的最高位。



1. 带进位循环右移指令 RCR

格式：RCR 操作数，（移位次数）COUNT

功能：对给定的操作数，右移COUNT次，每次移位时，最低位移入CF中，并把CF中的值移入空处的最高位。



六：转移指令

JMP 无条件转移 JZ（JE）结果为0转移 JNZ（JNE）结果不为0转移

JO 结果溢出转移 JNO 结果不溢出转移 JS 结果为负数转移

JNS 结果为正数转移 JP 结果中有偶数个1转移 JNP 结果中有奇数个1转移

无符号数

JC 结果<0转移 JNC 结果≥0转移

JA 结果>0转移 JNA 结果≤0转移

有符号数

JL 结果<0转移 JNL 结果≥0转移 JCXZ，CX=0转移

JG 结果>0转移 JNG 结果≤0转移

七：循环指令

LOOP指令

格式：LOOP 指令标号

等价于：DEC CX

JNZ 指令标号

例：将内存中1000H开始的500个字节数据复制到2000H开始的缓冲区中。

解：MOV [2000H]，[1000H] × MOV SI，1000H

MOV BL，[1000H] √ MOV DI，2000H

MOV [2000H]，BL MOV CX，500

MOV BL，[1001H] NEXT1：MOV BL，[SI]

MOV [2001H]，BL MOV [DI]，BL

MOV BL，[1002H] INC SI

MOV [2002H]，BL INC DI

LOOP NEXT1↔DEC CX

JNZ NEXT1

八：子程序

过程名 PROC NEAR/FAR

程序体

RET（返回程序）

过程名 ENDP

九：处理器控制指令

1. 标志处理指令
2. CLC CF置0 （2）STC CF置1 （3）CMC CF取反

（4）CLI IF置0 （5）STI IF置1

2.处理器控制

HLT 停机 WAIT 等待 ESC 换码 NOP 空操作 LOCK 封锁

1. ：汇编语言

一：汇编语言的格式

1. 指令语句←操作指令

[标号]（可选项） 指令助记符，操作数；注释

二：伪指令：对程序进行说明和定义，不直接产生结果

格式：[符号名] 伪指令符，操作数；注释

三：汇编语句表达式

1. 常量

|  |
| --- |
| 11H |
| 12H |
| 33H |
| 44H |
| 55H |
| 55H |
| 55H |
| 55H |

1. 变量 表达式项给变量或指定存储单元赋初值 A1

（1）数值表达式

数据定义伪指令，为一个或连续的存储单元赋初值 B1

DB 字节 DW 字 DD 双字

格式：[变量名] 数据定义伪指令 表达式1 C1

A1 DB 11H，12H

B1 DW 3344H

C1 DD 5555 5555H

（2）字符串表达式

字符串必须用单引号引起来，字符串中的字符以ASCⅡ码形式存放在相应的存储单元

|  |
| --- |
|  |
|  |
|  |
| 41H |
| 42H |

D1 DB ‘123’ D1

E1 DB ‘AB’

（3）N DVP（表达式）

注：N为重复因子，表示定义了N个重复数据存储单元 E1

例：BUF1 DB 100 DVP（0）

定义：以BUF1为首地址，100个字节存储单元，初值为0

三

1. 等价伪指令

格式：符号名 EQU 表达式

功能：为常量、表达式及其他符号定义一个等价的符号名，但不申请存储单元

1. 等号伪指令

格式：符号名 = 表达式

功能：为常量、表达式及其他符号定义一个等价的符号名，并且可以重复定义

例：COST1=25 COST1=COST1+3

1. 段定义伪指令

段名 SEGMENT

。 注：段名，为该段起的名字，用来指出汇编语言程序为该段分配的

。 存储区起始位置。

。

段名 ENDS

1. 假定伪指令

格式：ASSUME 段寄存器名 段名[，段寄存器名 段名]

功能：建立段寄存器与段之间的对应关系，一般出现在代码段中

说明：对于CS段寄存器，它不仅把对应的段分配给CS寄存器，并且把该段的段基址装入CS寄存器中。

对于其他段寄存器（DS，ES），只是指定把某个段分配给哪个寄存器，并没有把段基址装入相应的段寄存器中。

CODE SEGMENT

ASSUME DS：DATA，CS：CODE

MOV AX，DATA

MOV DS，AX DATA首地址送DS中

CODE ENDS

1. TITLE伪指令

为源程序定义一个标题

TITLE This is a program

1. 结束伪指令 END

格式：END 起始位置

功能：说明源程序已结束，并指明程序运行的起始位置

四：DOS系统功能调用

一般步骤：

①设置调用功能的入口参数

②把要调用的功能号放入AH寄存器中

③INT 21H指令自动转入中断子程序

④相应的中断子程序运行完毕得到出口参数

1. 单字符输入（1号调用）

格式：MOV AH，1

INT 21H

功能：从键盘输入字符的ASCⅡ码送入AL中，并显示出来。

1. 单字符显示（2号调用）

格式：MOV DL，待显示字符的ASCⅡ码

MOV AH，2

INT 21H

功能：将DL中的字符送显示器显示。

1. 结果调用（4CH号调用）

格式：MOV AH，4CH

INT 21H

1. 显示字符串（9号调用）

格式：MOV DS，待显示字符串的段基址

MOV DX，待显示字符串偏移地址

MOV AH，09H

INT 21H

功能：将“＄”结尾字符串送显示器显示。

五：综合举例

例1：设计一个程序将下列字符显示在屏幕上

China is our country！

解： TITLE Display a string

DATA SEGMENT

Outbuf DB 0DH,0AH,‘China is our country’＄’

DATA ENDS

CODE SEGMENT

ASSUME DS：DATA，CS：CODE

Start：MOV AX，DATA

MOV DS，AX

LEA DX，outbuf

MOV AH,09H 调用9号功能

INT 21H

MOV　AH，4CH 结果调用

INT 21H

CODE ENDS

END Start 结束

|  |
| --- |
| 56H |
| ？ |
| ？ |
| 34H |
| 12H |

例：程序中将要使用3个变量X1、X2和X3，其中，X1是长度为单字节变量，X2是双字节变量，X3也是一个双字节变量，X1的初始值变量为56H，X2不设定初始值，X3设的初始值为1234H

解： DATA SEGMENT X1

X1 DB（字节） 56H X2

X2 DW（字） ？

X3 DW 1234H X3

DATA ENDS

例：设圆的半径r=5，圆周率π=3，设计一个计算圆周长的程序，并将计算结果存入变量Length中。

解： Title Length of a circle

|  |
| --- |
| 5 |
| ？ |
| ？ |

PI EQU 3

DATA SEGMENT r

r DB 5 length

Length DW ？

DATA ENDS

CODE SEGMENT

ASSUME CS：CODE， DS：DATA

Start：MOV AX，DATA

MOV DS，AX

MOV AL，r

MUL PI （×）

MOV BL，PI

MUL BL ；注释：（AL）×（BL）→（AX）

MOV BX，2

MUL BX ；注释：（AX）×（BX）→DX：AX

MOV Length，AX ；注释：2πr→Length

MOV　AH，4CH 结果调用

INT 21H

CODE ENDS

END Start 结束

1. 存储系统
2. ：存储器概述

一：存储器分类

1. 按存储介质分类

①磁表面存储器 剩磁效应 采用磁性材料

②半导体存储器 半导体器件

补充：光盘 聚酯

1. 按存储方式分类

①随机存储器

②顺序存储器

1. 按读写功能分类

①只读存储器（ROM）

②随机读写存储器（RAM）

1. 按信息可保存性分类

①非永久记忆存储器（RAM）

②永久记忆存储器（ROM）

1. 按在计算机系统中的作用分类

主存（内存）、辅存（外村）、高速缓存、控制存储器（存放微指令）

二：存储器分级结构

CPU中的寄存器

高速缓存（cache）

主存

辅存

三：技术指标

存储容量 1KB（2^10） 1MB（2^20） 1GB（2^30） 1TB（2^40）

存取时间

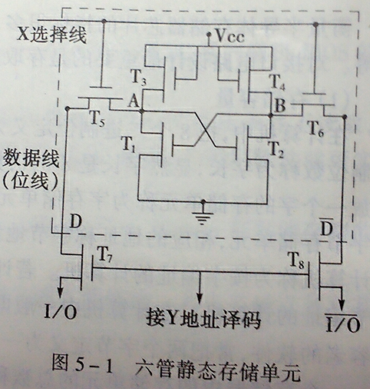
存取周期

存储器带宽

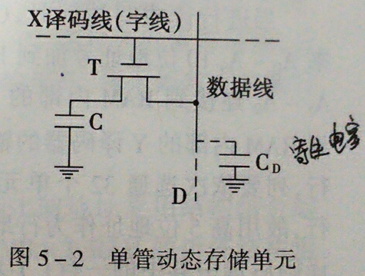
第二节：半导体存储器

一：随机存取存储器

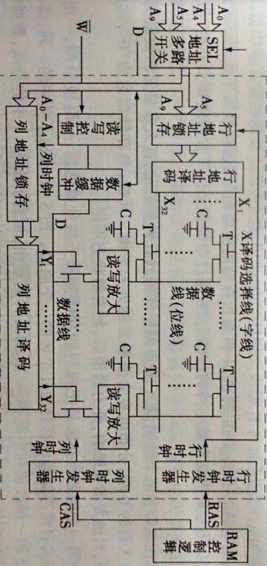
1.静态RAM（使用6个MOS管）

双稳态电路

1. 单管动态存储单元



二：RAM的结构



RAM采用阵列结构

优点：①减少对外引脚数

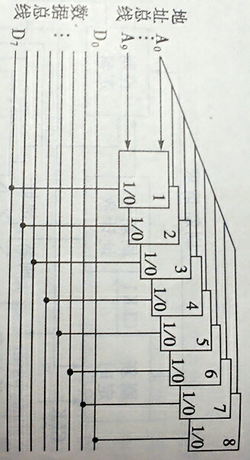
②便于刷新充电（动态RAM）

三：动态RAM的刷新

四：RAM的扩展

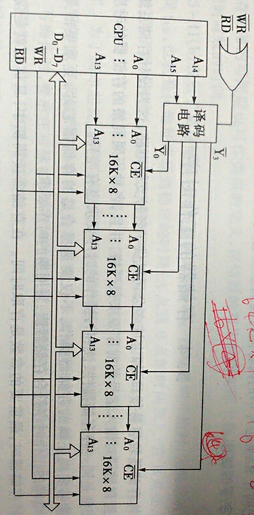
1. 位扩展

1K×1位→1K×8位



1. 字扩展

1K×1位→4K×8位

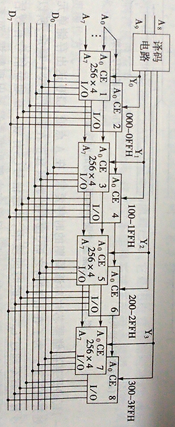


1. 字位同时扩展

1K×1位→1K×8位

↓

4K×8位



共4×8=32个1K×1位

五：存储器的工作时序

1. 读周期

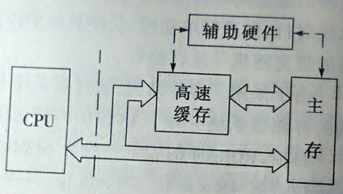
读周期 = 读取时间 + 读操作恢复时间

1. 写周期

写周期 = 地址建立时间 + 写脉冲宽度 + 写操作恢复时间

1. ：高速缓冲存储器（cache）

一：cache - 主存存储层次



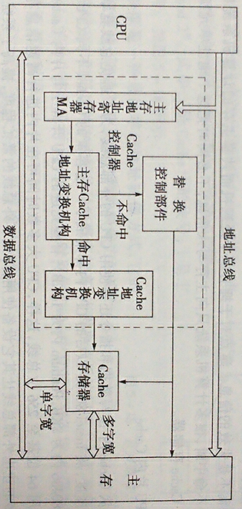
1. 使用cache的必要性

由动态RAM构成的主存比CPU慢一个数量级，整个计算机系统效率低下，在CPU与主存之间增加一个小容量的速度与CPU相当的双极性静态RAM作为cache是必要的。

1. 使用cache的可行性

根据程序局部性原理（即对局部范围地址频繁访问，而对此范围以外的地址很少访问）把正在执行指令地址附近的一部分指令或数据装入cache中，供CPU在一段时间内使用是可行的。

二：cache的基本工作原理



三：cache的读写操作

1. 读操作

①如果要读的指令或数据已在cache中，直接从cache中读取

②如果要读的指令或数据不在cache中，CPU直接从主存中读取，同时把主存中此地址附近的数据块放入cache中。

2.写操作

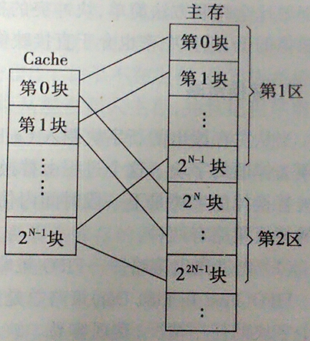
①通写法：cache和主存同时写（修改）

②回写法：每次只对cache进行写（修改）并作标记，当cache中此修改内容被替换出去cache时，才写回主存。

四：地址映像

1. 直接映像

每个主存地址映像到cache中的指定地址

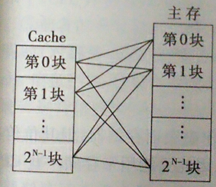


优点：简单，实现容易

缺点：易冲突，cache存储空间没有充分利用。

1. 全相联映像

主存中的每一个字块可以映像到cache中的任何字块位置

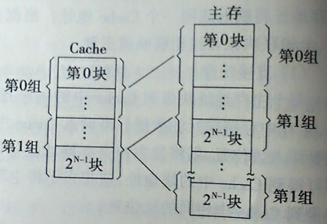


优点：cache存储空间充分利用，冲突少

缺点：查找空闲cache字块由硬件实现，复杂

1. 组相联映像

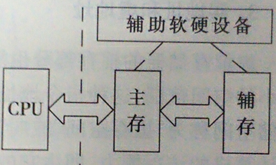
把cache和主存分成若干组，每组由若干字块组成，cache中的组与主存中的组之间采用直接映像，组内之间采用全相联映像。



五：替换策略

1. 先进先出策略—FIFO策略
2. 近期最少使用策略—LRU策略
3. ：虚拟存储器

一：主-辅存存储层次



二：虚拟存储器基本概念

虚拟存储器：主存和辅存的地址空间统一编制

主-辅存存储层次与cache-主存存储层次

相同之处：①都是基于程序局部性原理

②地址映像和替换策略相同

不同之处：①cache-主存换入换出由硬件完成，主存-辅存换入换出由操作系统完成

②主-辅存存储层次未命中时，性能损失远大于cache-主存层次

三：实地址：主存单元的地址

虚地址：指令地址

1. ：输入输出的接口技术
2. ：I/O接口

一：基本概念

I/O接口：CPU和I/O设备之间的接口逻辑及有关软件

二：接口的功能

1. 寻址功能
2. 缓冲功能
3. 数据格式转换
4. A/D或D/A转换
5. 电平转换
6. 由接口实现CPU与外设的不同传输方式
7. 联络功能

三：接口的分类

1. 按数据传输的基本方式

串行通信

并行通信

1. 按I/O数据传输方式

直接程序传输

程序查询

中断程序

DMA接口

1. 接口的总线标准

ISA总线接口（8位或16位）

EISA总线接口（32位）

PCI

STD总线接口（工业标准）

1. ：I/O端口

一：I/O端口

接口中可供CPU访问的寄存器、三态门等硬件逻辑

分为数据端口、状态端口、控制端口

二：I/O端口编址方法

1. 与存储器统一编址：访问存储器和I/O端口采用同样指令。

优点：指令系统简单。

缺点：I/O端口使用存储地址，造成存储空间不能充分利用。

1. I/O端口独立编址：访问存储器和I/O端口采用不同指令。

优点：存储空间可充分利用。

缺点：指令系统相对复杂。

1. ：CPU和外设之间的数据传输基本方式

一：直接程序传输（又称无条件传输）

CPU与外设之间利用程序传输时，认为外设已准备好，可以直接进行传输。

二：程序查询传输（又称有条件传输）

CPU与外设之间利用程序传输之前，先查询外设的状态，若满足条件，则进行传输，否则等待。

三：中断程序传输

CPU在每个指令周期的最后一个T周期，检测可屏蔽中断请求引脚INTR，若INTR=1，IF=1，则响应中断。

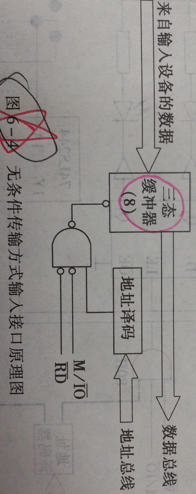
四：DMA方式

直接存储器存取方式：外设利用专门硬件电路直接和存储器进行高速数据传输，在传输过程中无需CPU干预。

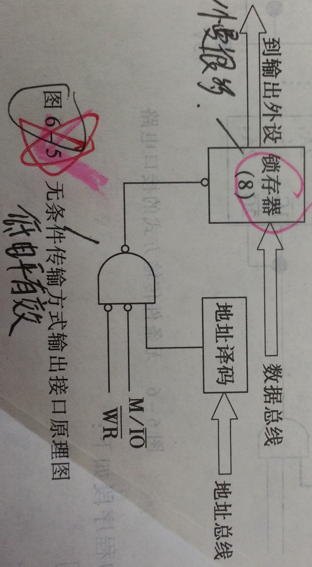
1. ：输入输出的接口技术

一：直接程序传输

1. 输入（外设→CPU）



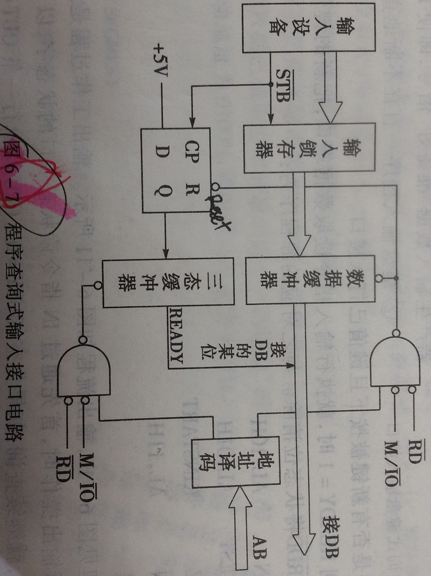
2.输出（CPU→外设）

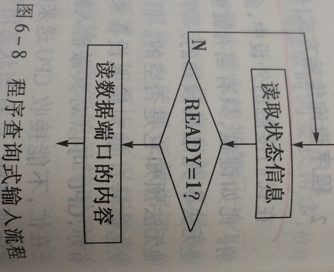


因为外设比CPU慢很多，所以使用锁存器

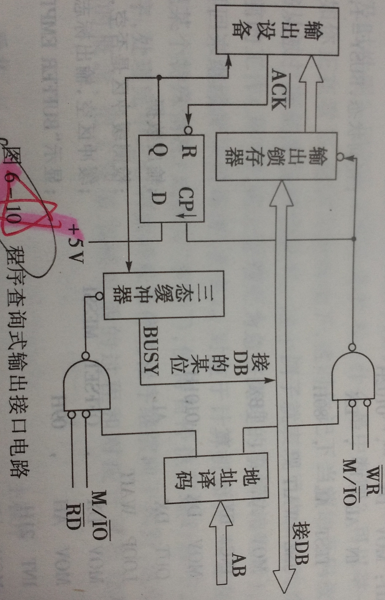
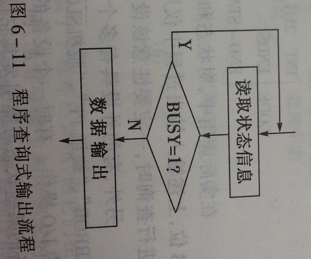
二：程序查询传输

1. 输入（外设→CPU）

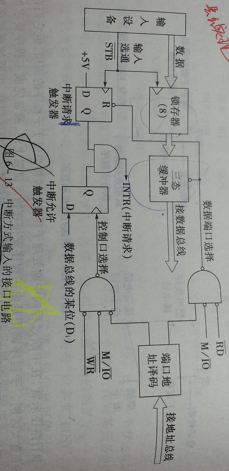




2.输出（CPU→外设）

三：中断程序传输



四：DMA方式

直接存储器存取方式：通过DMA控制器实现（DMA8237）

1. ：I/O端口地址设计

例1：设计一个端口地址为2FBH的译码电路

解： A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0

0 0 1 0 1 1 1 1 1 0 1 1

2 F B H

例2：设计一个片内拥有8个端口的接口电路的I/O地址译码器电路，其端口地址为310H-317H。

解： A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0

0 0 1 1 0 0 0 1 0 0 0 0

3 1 0 H

0 0 1 1 0 0 0 1 0 0 0 1

3 1 1 H

……

0 0 1 1 0 0 0 1 0 1 1 1

3 1 7 H

例3：某微机应用系统，包含8个接口电路，每个接口拥有4个端口，试设计地址译码电路，其地址分配在200H—21FH

解：A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0

0 0 1 0 0 0 0 0 0 0 0 0 200H

0 0 1 0 0 0 0 0 0 0 0 1 201H

……

0 0 1 0 0 0 0 1 1 1 1 1 21FH

设计思路：

①用A4A3A2产生8个片选，选择8个接口电路

②A1A0寻址4个端口

C B A

0 0 0 →

0 0 1 →

0 1 0 →

0 1 1 →

1 0 0 →

1 0 1 →

1 1 0 →

1 1 1 →

1. ：中断技术
2. ：中断的基本概念

中断

主程序

中断服务子程序

断点：被中断的主程序的位置

断点保护：压入堆栈

断点恢复：弹出堆栈

现场保护

现场恢复

中断源

中断请求：外设 → CPU

中断响应：外设 ← CPU

1. ：中断的典型应用
2. CPU与外设传输
3. 实时处理
4. 故障处理
5. 系统调度
6. 多机通信
7. ：

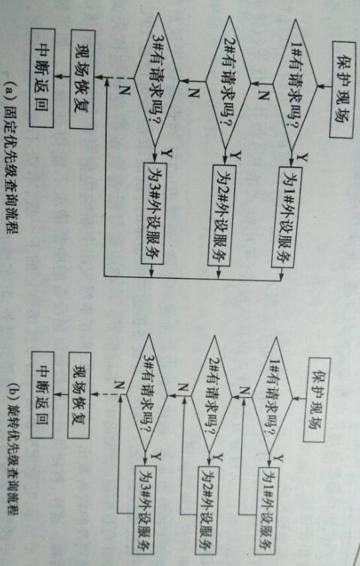
一：基本功能

1. 能实现中断响应和返回
2. 中断优先级排序
3. 能实现中断嵌套

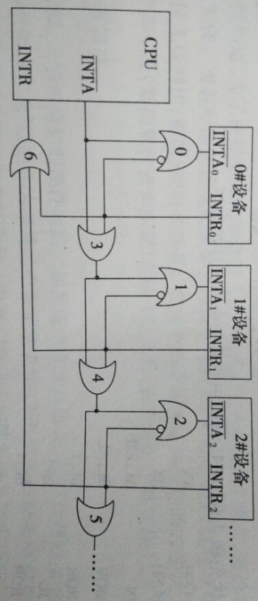
高优先级可以中断低优先级，同级或低级中断不能中断当前中断

二：中断优先级

1. 软件查询法
2. 固定优先级
3. 等优先级（循环优先级）



1. 硬件判优法
2. 简单硬件判优



1. 专用硬件方式
2. ：8086中断操作

一：8086中断分类

1. 硬件中断

可屏蔽中断（外设 → CPU）

不可屏蔽中断（外设 → CPU）

1. 软件中断

中断指令

除法错中断（0型）

单步中断（1型）

断点中断（3型）

溢出中断（4型）

二：中断类型码，中断向量，中断向量表

1. 中断类型码（0—255）：每个中断源分配一个中断类型码
2. 中断向量：中断服务子程序在存储器存放的起始位置由段基址和段内偏移量组成
3. 中断向量表：系统中所有中断向量按一定规律排列成一个表存放在内存0段的0000H—03FFH
4. 中断服务程序的转移：中断源的中断类型码左移两位（即乘4），就得到该中断源的中断向量在中断向量表中存放的起始位置
5. 中断向量表的初始化
6. 由BIOS提供的中断服务
7. 由DOS提供的中断服务

例：设某中断类型号为72H的用户程序的入口地址为USEINT，请初始化

解：72H×4 = 1C8H

MOV AX，0

MOV DS，AX

MOV BX，01C8H

MOV AX，OFFSET USEINT

MOV [BX]，AX

MOV AX，SEG USEINT

INC BX

INC BX

MOV [BX]，AX

DOS子功能调用

1. 从中断向量表中读出中断向量（35H子功能）

MOV AL，中断类型码

MOV AH，35H

INT 21H

结果：ES：BX ← 中断向量（ES：段基址，BX：段内偏移地址）

1. 把中断向量写入中断向量表

MOV AX，中断向量的段基址 SEG USEINT

MOV DS，AX

MOV DX，中断向量的段内偏移量 OFFSET USEINT

MOV AL，中断类型码（72H）

MOV AH，25H

INT 21H

1. ：中断响应过程和时序

一：硬件中断的响应过程

1. INTR=1，IF=1（外设 → CPU）
2. CPU → 外设
3. 外设送中断类型码 → CPU
4. CPU将中断类型码暂存
5. 将标志寄存器内容压入堆栈
6. 将IF和TF清零
7. 断点保护
8. 根据中断类型码（左移2位），在中断向量表中找出中断向量，再在存储器中取出中断服务子程序并运行
9. 断点恢复
10. 标志寄存器恢复（弹出堆栈）
11. 返回主程序

二：8086中断响应的流程

↓

内部中断？（软件中断）

↓

NMT？（硬件中断）

↓

INTR？（硬件中断）

↓

TF？（单步中断）

三：硬件中断的时序

第一个中断响应周期（INTR）

检测INTR 以确认

第二个中断响应周期

外设送中断类型码给CPU

四：软件中断

1. 现场保护
2. 断点保护
3. 根据中断类型码得到中断向量，从而调用中断处理子程序并执行
4. 断点恢复
5. 现场恢复
6. 返回主程序
7. ：中断控制器Intel 8259A

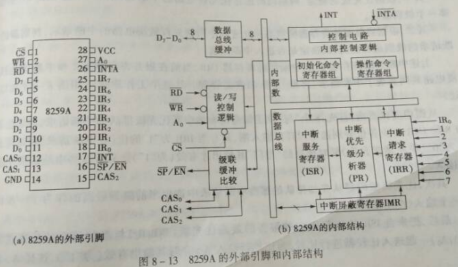
一：8259A的主要功能和结构

1. 主要功能

（1）1片8259A可以管理8级中断，9片8259A级联可以处理64级中断。

1. 任一级中断可以中止，允许。
2. 通过编程，设置中断类型码和工作方式

二：8259A内部结构



①中断请求寄存器IRR

D7 D6 D5 D4 D3 D2 D1 D0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

IR7 IR6 IR5 IR4 IR3 IR2 IR1 IR0

为1：服务

②中断屏蔽寄存器IMR

D7 D6 D5 D4 D3 D2 D1 D0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

IR7 IR6 IR5 IR4 IR3 IR2 IR1 IR0

为1：屏蔽 为0：允许

③中断屏蔽寄存器ISR

D7 D6 D5 D4 D3 D2 D1 D0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

IR7 IR6 IR5 IR4 IR3 IR2 IR1 IR0

为1：服务 为0：不服务

二：工作过程

三：8259A的中断管理

1. 中断优先级管理

①固定优先级 IR0>IR1>.....>IR7

②循环优先级：一旦某中断源的请求被响应后它就变为最低优先级，它的下一级则上升为最高优先级。

③特殊循环优先级

如指定IR2为最低，IR3就为最高

④查询排序优先级

1. 中断结束管理

①自动结束方式

②一般结束方式

③特殊结束方式

1. 中断嵌套管理

①完全嵌套方式

②特殊全嵌套方式

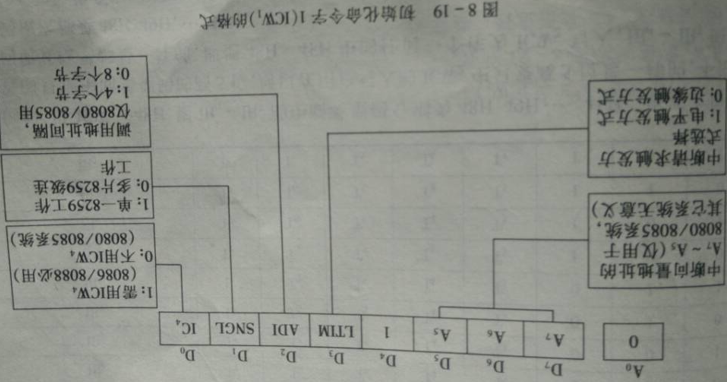
1. 中断屏蔽你管理

①一般屏蔽方式

②特殊屏蔽方式

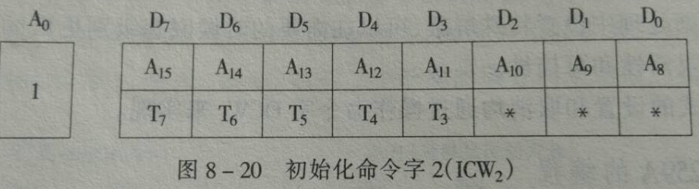
四：8259A的编程

1. 初始化命令字
2. 初始化命令字1（ICW1）



1. 初始化命令字2（ICW2）

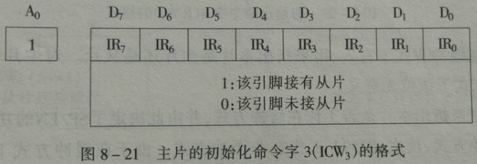
用来设置中断类型码



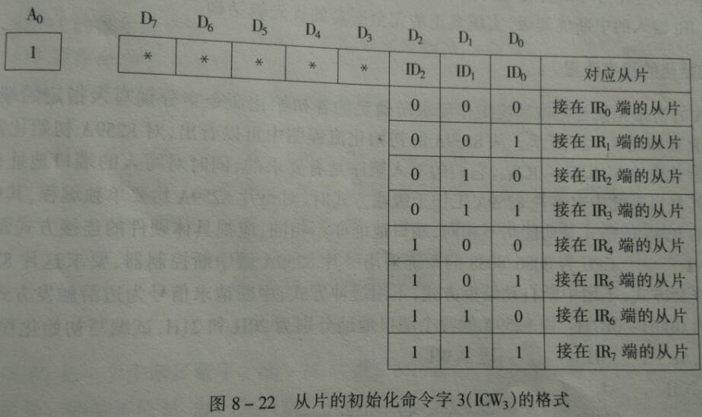
1. 初始化命令字3（ICW3）

有多片8259A级联时才需要写入ICW3。级联系统中的所有8259A都必须进行独立编程，以便设置各自的工作状态。作为主设备的8259A一般设置为特殊的全嵌套方式，作为从设备的8259A一般设置为完全嵌套方式。

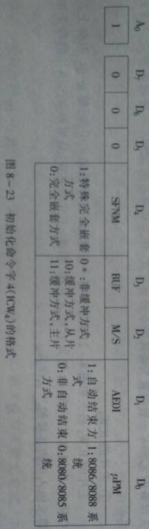
①主片的ICW3



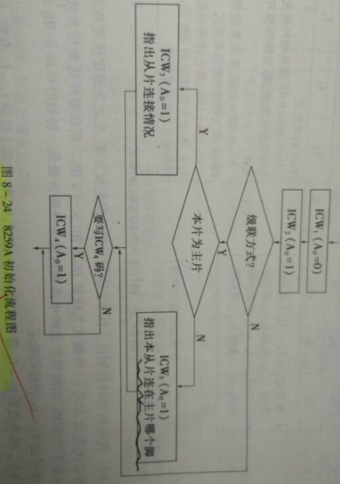
②从片的ICW3



（4）初始化命令字4（ICW4）



2.初始化的程序流程



例题：一个CPU为8086/8088的系统采用一片8259A做中断控制器，要求这片8259A工作在全嵌套方式，不用中断自动结束方式，不用缓冲方式，中断请求信号为边沿触发方式，中断类型码为0B0H—0B7H。该8259A的两个端口地址分别为20H和21H，试编写程序。

解： MOV AL，13H ；送ICW1

OUT 20H，AL

MOV AL，0B0H ；送ICW2，写入中断类型码的高5位

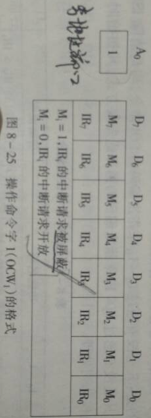
OUT 21H，AL

MOV AL，01H ；送ICW4

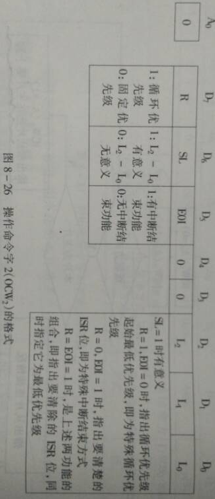
OUT 21H，AL

3.操作命令字

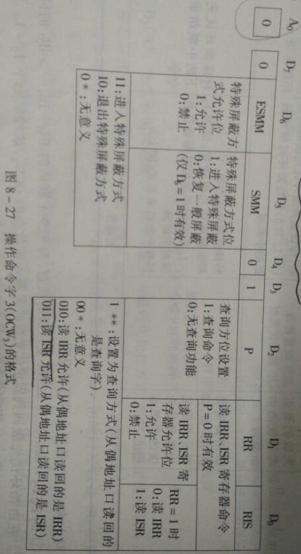
1. 操作命令字1（OCW1）



1. 操作命令字2（OCW2）



1. 操作命令字3（OCW3）



查询方式工作过程：

①关闭系统中断：执行一条CLI指令

②设置查询命令字：OCW3为OCH

③读取查询字：

