专业课-微机原理与接口

目录

第一章 微型计算机的基础知识

1.1、常见的数制

1.1.1、数制的分类

数制	基数	权值	数码
二进制	2	以 2 为底的幂	**0，1**
八进制	8	以 8 为底的幂	**0，1，2，3，4，5，6，7**
十进制	10	以 10 为底的幂	**0，1，2，3，4，5，6，7，8，9**
十六进制	16	以 16 为底的幂	**0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F**

1.1.2、数制的转换

二进制转任意进制	十进制转任意进制
八进制：　每 3 位二进制表示 1 位　八进制 十六进制：每 4 位二进制表示 1 位十六进制	整数部分：除基取余法，逆序排列 小数部分：乘基取整法，顺序排列

1.2、常见的码制

ASCII	BCD
ASCII 码是美国信息交换标准代码的简称，用于给西文字符编码，这种编码由 7 位二进制数组合而成，可以表示 128 种字符，包括英文字母的大小写、数字、专用字符、控制字符等。 换行（0AH）、回车（0DH）、空格（20H） 数字：30H~39H 大写字母：41H~5AH 小写字母：61H~7AH	BCD 码专门解决用二进制数表示十进数问题。 （1）压缩 BCD 码用一个字节表示两位十进制数，高 4 位的 0000~1001 表示 0~9，低 4 位的 0000~1001 表示 0~9。 （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。

1.3、数据的表示

1.3.1、机器数和真值

机器数	在机器中使用的连同数符一起数码化的二进制数	例如：01100110B、10011001B
真值	将带符号位机器数对应的真正数值称机器数的真值	例如：10000001B=-0000001=-1

1.3.2、机器数表示范围

类型	字长	最小值（补码）	最大值（补码）
无符号数	8 位	0000 0000B （0）	1111 1111B （255）
	16 位	0000 0000 0000 0000B （0）	1111 1111 1111 1111B （65535）
有符号数	8 位	原码范围：-127~+127（+0、-0） 反码范围：-127~+127（+0、-0） 补码范围：-128~+127（+0、-0==-128）	
	16 位	原码范围：-32767~+32767（+0、-0） 反码范围：-32767~+32767（+0、-0） 补码范围：-32768~+32767（+0、-0==-32768）	

1.3.3、原码&反码&补码

正数		负数	
原码	0000 0101B	**原码**	1010 1010B
反码	0000 0101B	**反码**	1101 0101B
补码	0000 0101B	**补码**	1101 011X B 0110

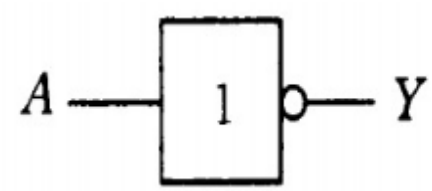
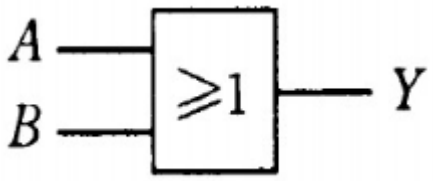
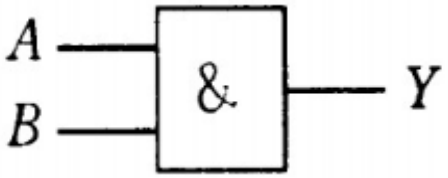
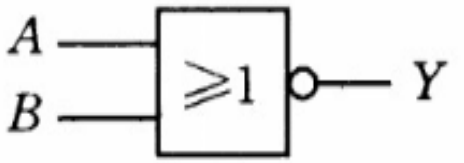
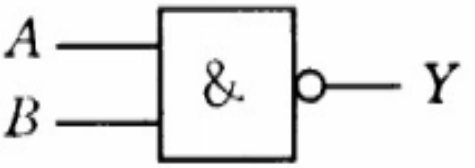
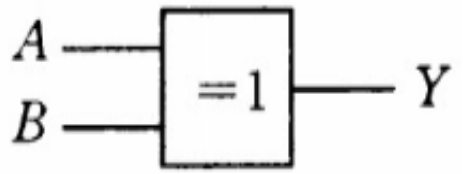
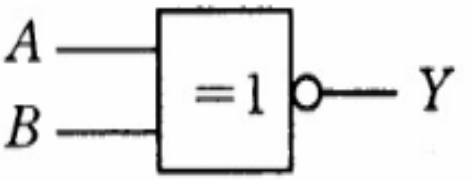
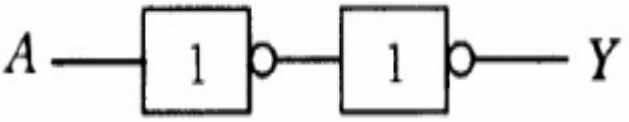
1.4、布尔代数

或运算：各变量全为 0 时则结果必为 0，变量中有一为 1 时则结果为 1。

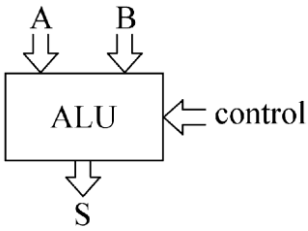
与运算：各变量全为 1 时则结果必为 1，变量中有一为 0 时则结果为 0。

第二章 微型计算机的组成电路

2.1、逻辑电路

1	非门		$Y=\overline{A}$
2	或门		$Y=A+B$
3	与门		$Y=A\times B$
4	或非门		$\overline{Y=A+B}$
5	与非门		$\overline{Y=A\times B}$
6	异或门 (异门)		$Y=\overline{A}B+A\overline{B}$
7	异或非门 (同门)		$Y=AB+\overline{A}\overline{B}$
8	缓冲器 (变阻器)		$Y=A$

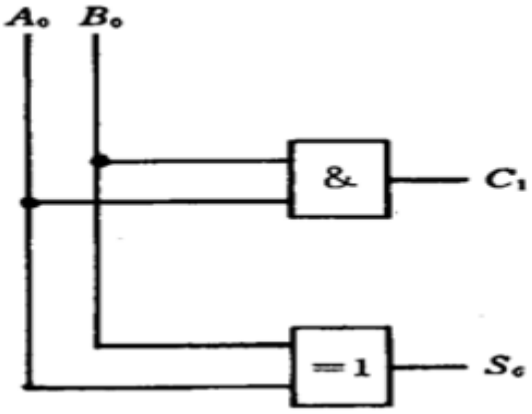
2.2、算术逻辑单元



算术逻辑单元（ALU）既能进行二进制的四则运算，也能进行布尔代数的逻辑运算。

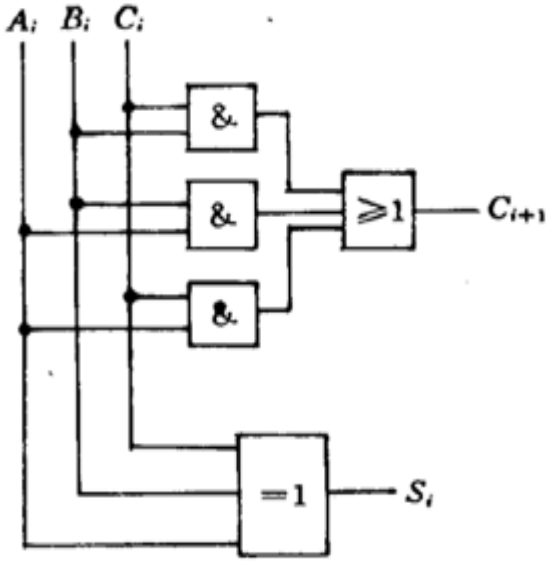
2.2.1、半加器电路及真值表

A_0	B_0	C_1	S_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

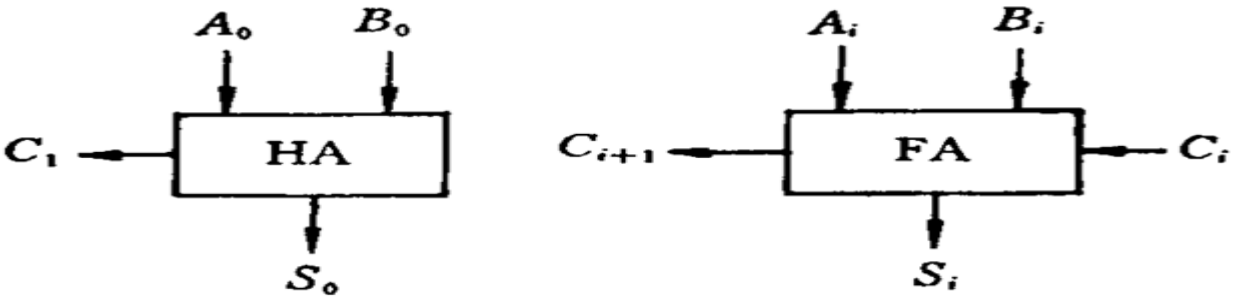


2.2.2、全加器电路及真值表

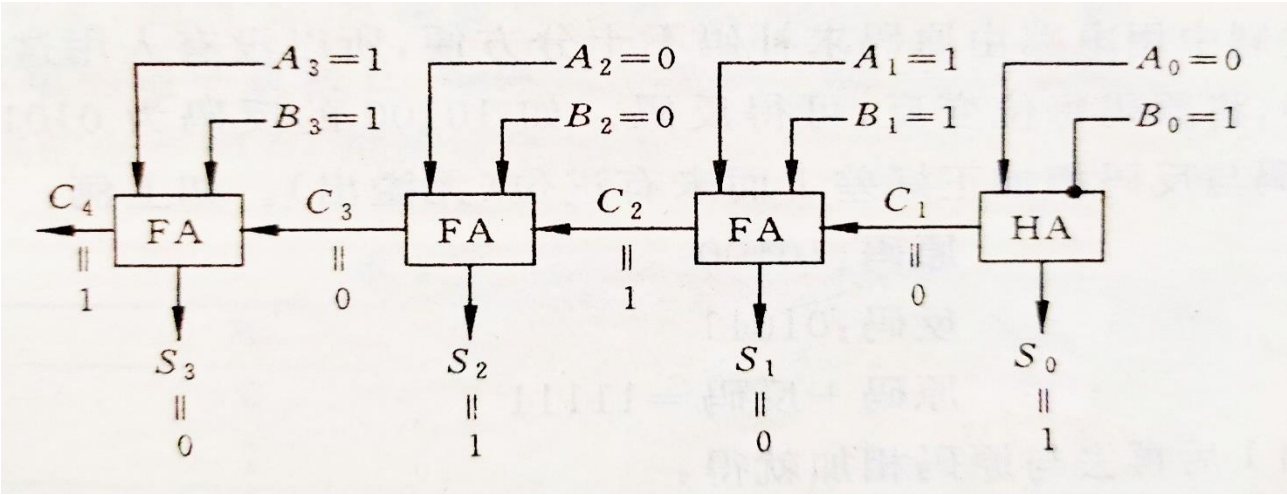
A_i	B_i	C_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



2.2.3、半加器和全加器符号

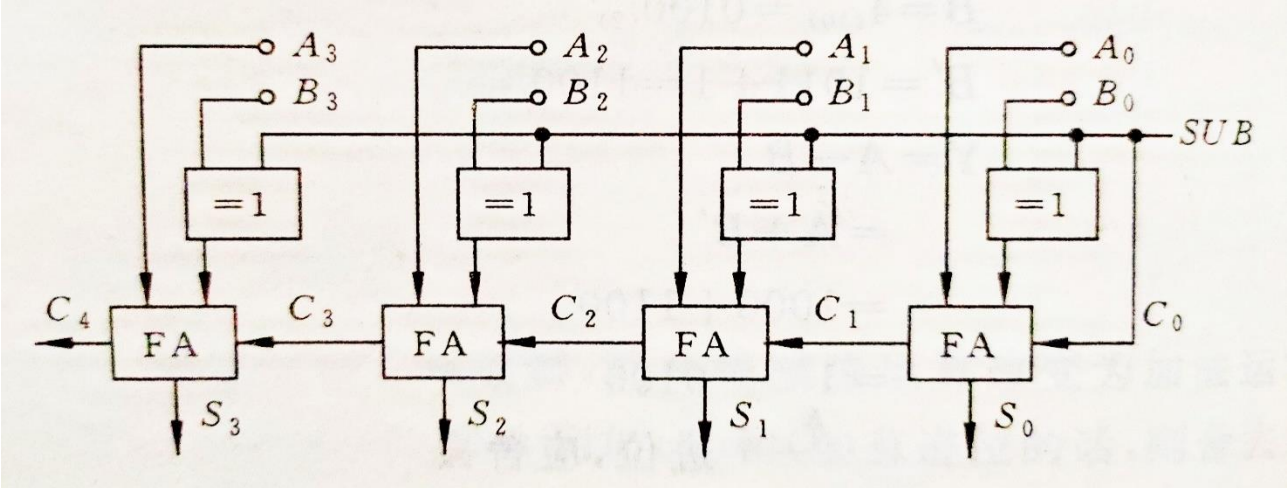


2.2.4、加法器电路的结构图



加法：$S = C_4S_3S_2S_1S_0$

2.2.5、补码加法器和减法器



加法：$S = C_4S_3S_2S_1S_0$　　减法：$S = S_3S_2S_1S_0$

2.3、触发器

2.3.1、RS 触发器

RS 触发器又称复位/置位触发器是组成其它触发器的基础，可以用与逻辑组成，也可以用或逻辑组成，其用途之一是构成“防抖动电路”。

2.3.2、JK 触发器

JK 触发器是数字电路触发器中的一种基本电路单元。JK 触发器具有置 0、置 1、保持和翻转功能，在各类集成触发器中，JK 触发器的功能最为齐全。

在实际应用中，它不仅有很强的通用性，而且能灵活地转换其它类型的触发器，由 JK 触发器可以构成 D 触发器和 T 触发器。

2.3.3、D 触发器

D 触发器是一个具有记忆功能的，具有两个稳定状态的信息存储器件，是构成多种时序电路的最基本逻辑单元，也是数字逻辑电路中一种重要的单元电路。

2.3.4、T 触发器

T 触发器是在数字电路中，凡在 CP 时钟脉冲控制下，根据输入信号 T 取值的不同，具有保持和翻转功能的电路，即当 T=0 时能保持状态不变，T=1 时一定翻转的电路。

2.4、寄存器

寄存器是由触发器组成的，一个触发器就是一个一位寄存器，由多个触发器可以组成一个多位寄存器。

常见的寄存器有：缓冲寄存器、移位寄存器、计数器、累加器

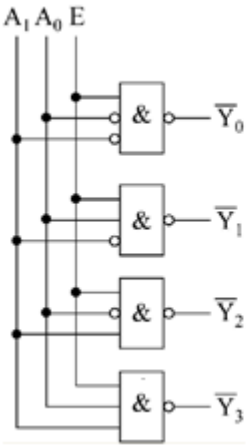
CX AX

2.5、三态门

计算机中的记忆元件由触发器组成，而触发器只有两个状态，即“0”态和“1”态，所以每条信号线上只能传送一个触发器的信息。如果要在一条信号线上连接多个触发器，而每个触发器可以根据需要与信号线连通或断开，当连通时可以传送“0”或“1”，断开时对信号线上的信息不产生影响，就需要一个特殊的电路加以控制，此电路即为三态输出电路，又称为三态门。

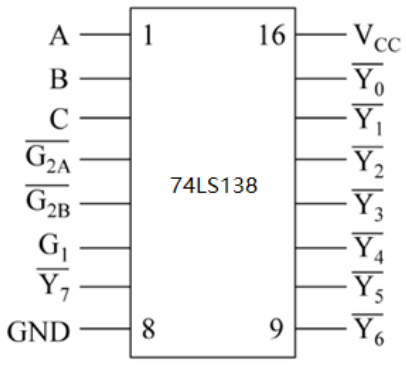
2.6、译码器

2.6.1、2-4 译码器



控制端	输入端		输出端			
E	A_1	A_0	$\overline{Y}_3$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$
1	0	0	1	1	1	0
	0	1	1	1	0	1
	1	0	1	0	1	1
	1	1	0	1	1	1

2.6.2、3-8 译码器



控制端			输入端			输出端							
G_1	$\overline{G}_{2B}$	$\overline{G}_{2A}$	C	B	A	$\overline{Y}_7$	$\overline{Y}_6$	$\overline{Y}_5$	$\overline{Y}_4$	$\overline{Y}_3$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$
1	0	0	0	0	0	1	1	1	1	1	1	1	0
			0	0	1	1	1	1	1	1	1	0	1
			0	1	0	1	1	1	1	1	0	1	1
			0	1	1	1	1	1	1	0	1	1	1
			1	0	0	1	1	1	0	1	1	1	1
			1	0	1	1	1	0	1	1	1	1	1
			1	1	0	1	0	1	1	1	1	1	1
			1	1	1	0	1	1	1	1	1	1	1

2.7、存储器

2.7.1、存储器的概念

存储器是计算机的重要部件，有记忆功能，用来存放指令代码和操作数。

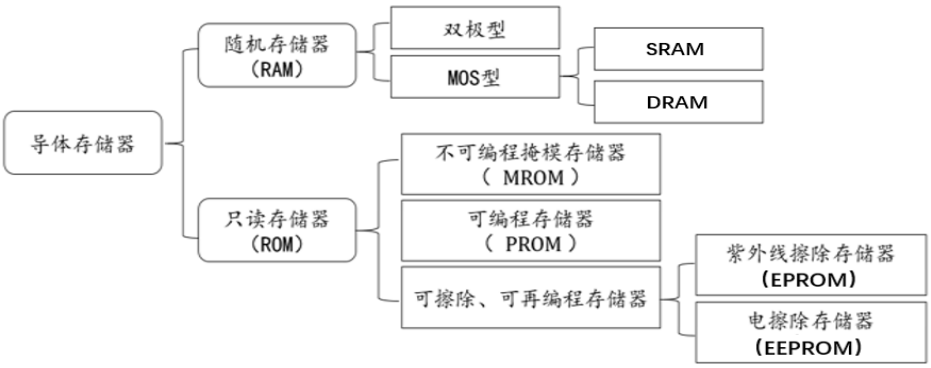
2.7.2、存储器的存储容量

存储容量	存储范围	地址码	存储单元
512B	000H~1FFH	9 位地址码	200H 个
1KB	000H~3FFH	10 位地址码	400H 个
2KB	000H~7FFH	11 位地址码	800H 个
4KB	000H~FFFH	12 位地址码	1000H 个
8KB	0000H~1FFFH	13 位地址码	2000H 个
16KB	0000H~3FFFH	14 位地址码	4000H 个
32KB	0000H~7FFFH	15 位地址码	8000H 个
64KB	0000H~FFFFH	16 位地址码	10000H 个
128KB	00000H~1FFFFH	17 位地址码	20000H 个
256KB	00000H~3FFFFH	18 位地址码	40000H 个
512KB	00000H~7FFFFH	19 位地址码	80000H 个
1MB	00000H~FFFFFH	20 位地址码	100000H 个

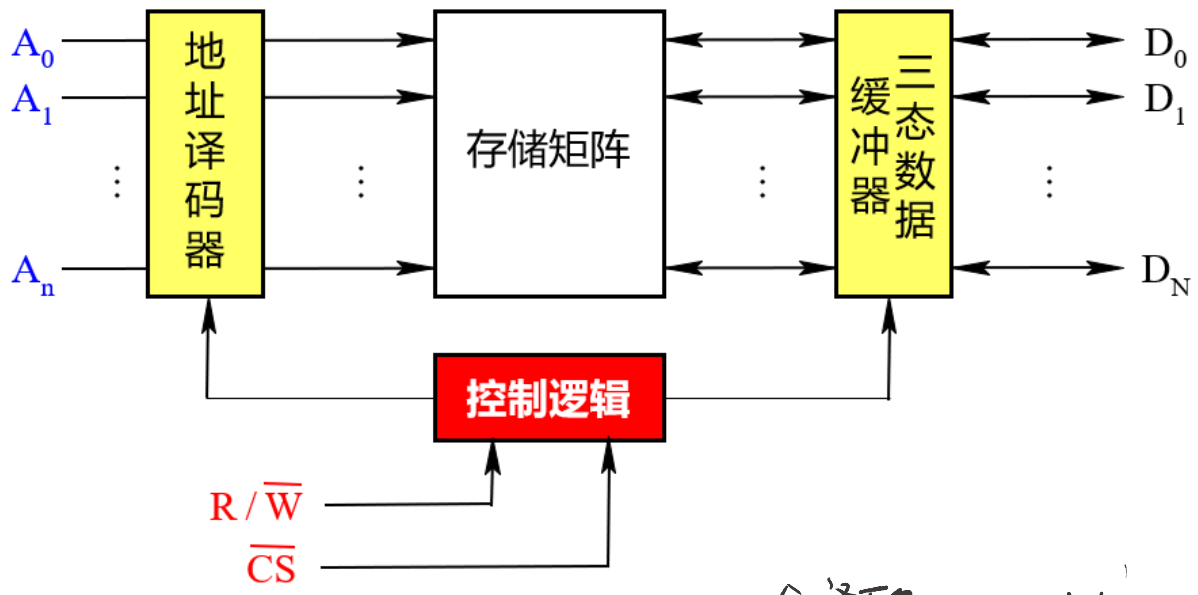
2.7.3、存储器的物理地址

物理地址 = 段地址 × 16 10H + 偏移地址

2.7.4、存储器的常见分类



2.7.5、存储器的基本结构



2.7.5.1、存储矩阵

每个基本存储电路存储一位二进制信息，这些存储电路有规则地组织起来，构成了存储体（存储矩阵）。不同存取方式的芯片，采用的基本存储电路也不同。

存储容量有两种表示方法：

①、位表示方法。以存储器中的存储地址总数与存储字位数的乘积表示。如 1K×4 位，表示该芯片有 1K 个单元，每个存储单元的长度为 4 个二进制位。

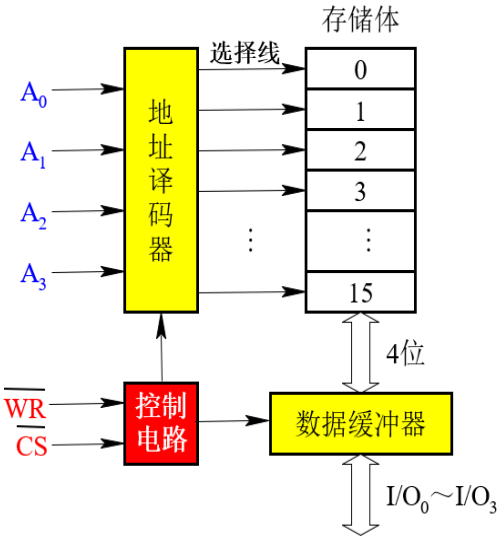
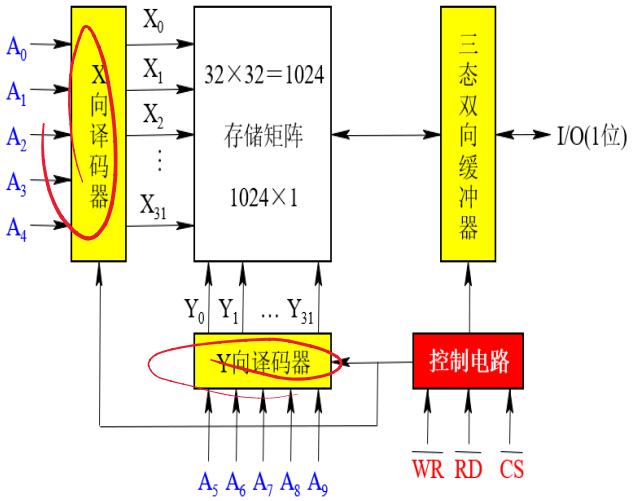
②、字节表示方法。以存储器中的单元总数表示。如 128B，表示该芯片有 128 个单元。

2.7.5.2、外围电路

外围电路主要包括地址译码电路、三态数据缓冲器电路、控制逻辑读/写电路。

2.7.5.3、译码方式

芯片内部的地址译码主要有两种方式：单译码方式、双译码方式。

单译码方式	双译码方式
单译码方式，只用一个译码电路对所有地址信息进行译码。译码输出的选择线直接选中对应的单元。一根译码输出选择线对应一个存储单元，故在存储容量较大、存储单元较多的情况下。	双译码方式，把 n 位地址线分为两部分，行选择线 X 和列选择线 Y，分别译码。每一根 X 选择线选择存储体中位于同一行的所有元素，每一根 Y 选择线选择存储体中位于同一列的所有元素，当某一单元的 X 线和 Y 线同时有效时，相应存储单元被选中。
	

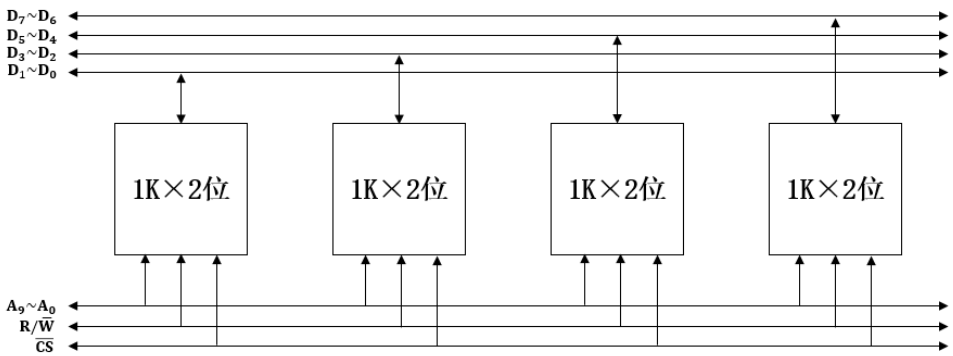
2.7.6、存储器的拓展技术

2.7.6.1、位拓展

描述：有一个1K×8位的存储器，由1K×2位的SRAM芯片构成，请设计存储系统，画出存储体的组成框图。

分析：存储体单元字数符合要求，但单元的位数较少时，需要进行位拓展。

答案：(1)$\frac{1K\times 8位}{1K\times 2位}=4$片；(2)$2^n$=1K，n=10，地址线为10根，接$A_9$~$A_0$；(3)数据线为8根，接$D_7$~$D_0$；



2.7.6.2、字拓展

描述：有一个4K×8位的存储器，由1K×8位的SRAM芯片构成，请设计存储系统，画出存储体的组成框图。

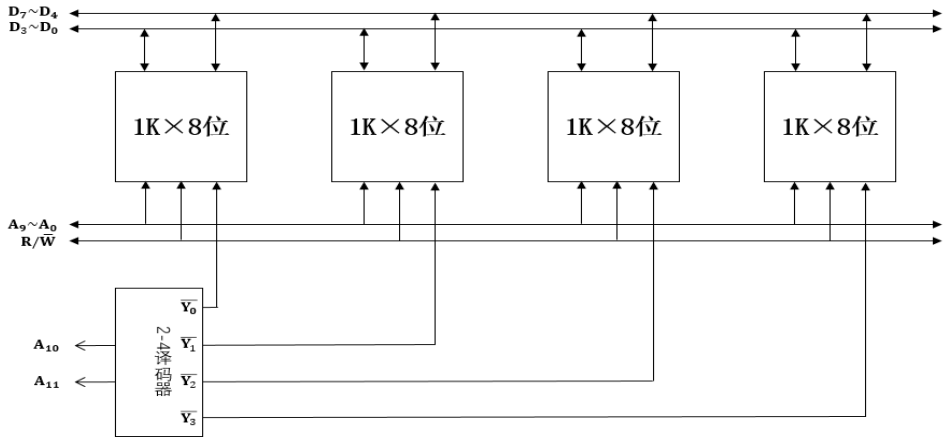
分析：存储体单元位数符合要求，但单元的字数较少时，需要进行字拓展。

答案：(1)$\frac{4K\times 8位}{1K\times 8位}=4$片；

(2)2^n=4K，n=12，总地址线为10根，接$A_{11}\sim A_0$；2^n=1K，n=10，片内地址线为10根，接$A_9\sim A_0$；

(3)片选需要$2-4$译码器，接$A_{11}\sim A_{10}$；

(4)数据线为8根，接$D_7\sim D_0$；



2.7.6.3、字位全拓展

描述：有一个16K×8位的存储器，由4K×4位的SRAM芯片构成，请设计存储系统，画出存储体的组成框图。

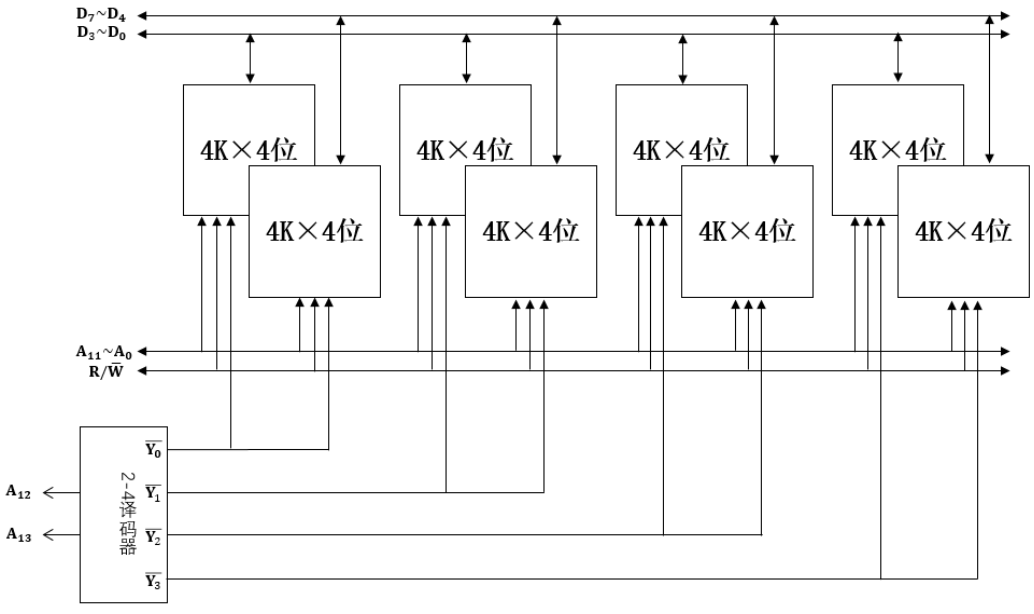
分析：存储体单元字数和位数均不符合要求，需要进行字位全拓展。

答案：(1)$\frac{16K\times 8位}{4K\times 4位}=8$片；一共4组，每组2片；

(2)2^n=16K，n=14，总地址线为14根，接$A_{13}\sim A_0$；2^n=4K，n=12，片内地址线为12根，接$A_{11}\sim A_0$；

(3)片选需要$2-4$译码器，接$A_{13}\sim A_{12}$；

(4)数据线为8根，接$D_7\sim D_0$；



2.7.7、随机存取存储器

2.7.7.1、SRAM

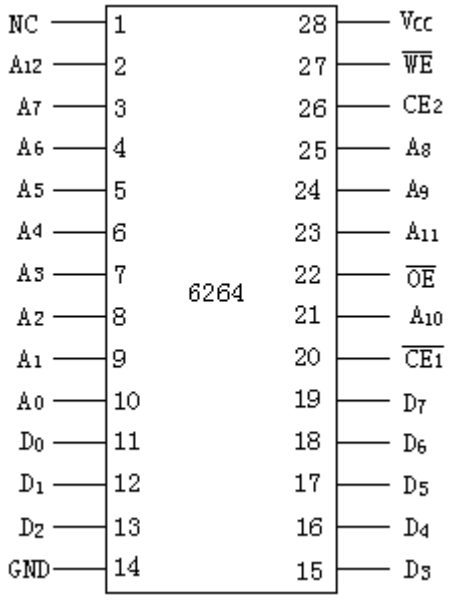
2.7.7.1.1、概述

SRAM 存储元件所用 MOS 管多，占硅片面积大，因而功耗大，集成度低，但是存取速度快，多用于小规模微机系统。

2.7.7.1.2、常见的芯片

类型	型号	容量	组织形式
SRAM	6232	32K	4K × 8bit
	6264	64K	8K × 8bit
	62128	128K	16K × 8bit
	62256	256K	32K × 8bit
	62512	512K	64K × 8bit

2.7.7.1.3、典型的芯片



SRAM 芯片-6264：

- 地址线：A_{12}~A_0 8K
- 数据线：D_7~D_0 8bit
- 片选 1 线：$\overline{CE_1}$
- 片选 2 线：CE_2
- 写入线：$\overline{WE}$
- 输出线：$\overline{OE}$

2.7.7.2、DRAM

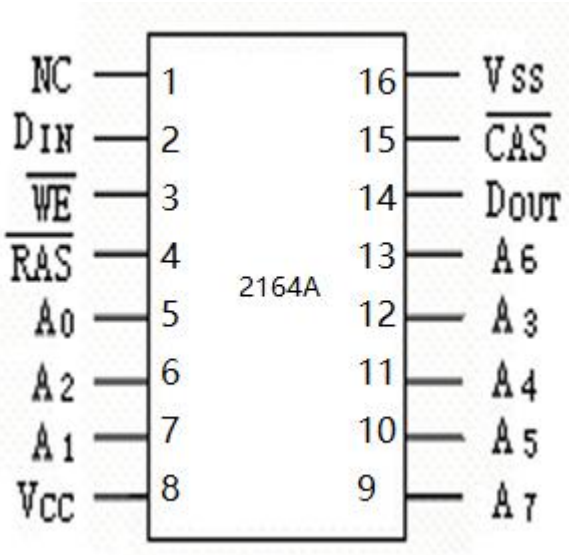
2.7.7.2.1、概述

DRAM 存储元件所用 MOS 管少，占硅片面积小，因而功耗小，集成度高，但是存取速度慢，多用于大规模微机系统。

2.7.7.2.2、常见的芯片

类型	型号	容量	组织形式
DRAM	2116A	16K	16K × 1bit
	2164A	64K	64K × 1bit
	4116	16K	16K × 1bit
	4164	64K	64K × 1bit

2.7.7.2.3、典型的芯片



DRAM 芯片-2164A：

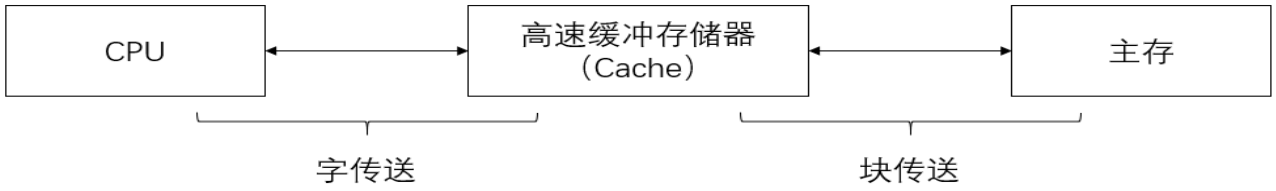
- **地址线：A_7~A_0**
- **数据输入线：D_{IN}**
- **数据输出线：D_{OUT}**
- **行地址选通：$\overline{RAS}$**
- **列地址选通：$\overline{CAS}$**
- **写入线：$\overline{WE}$**

2.7.8、高速缓冲存储器

2.7.8.1、概述

高速缓冲存储器是存在于主存与 CPU 之间的一级存储器，由静态存储芯片(SRAM)组成，容量比较小但速度比主存高得多，接近于 CPU 的速度，它的出现是为了解决主存与 CPU 之间速度不匹配的问题。

2.7.8.2、基本原理



2.7.8.3、地址映射

Cache 通过地址映射的方法确定主存块与 Cache 行之间的对应关系，确定一个主存块应该放到哪个 Cache 行中。

(1)、直接映射：将主存块存储到唯一的一个 Cache 行。

- 优点：直接映射最简单、不涉及替换策略问题、地址变换速度快
- 缺点：命中率较低、空闲空间无法利用

(2)、全相联映射：将主存块存储到任意的一个 Cache 行。

- 优点：命中率较高、存储空间利用率高
- 缺点：线路复杂、成本高、速度低

(3)、组相联映射：将主存块存储到唯一的一个 Cache 组的任意行。

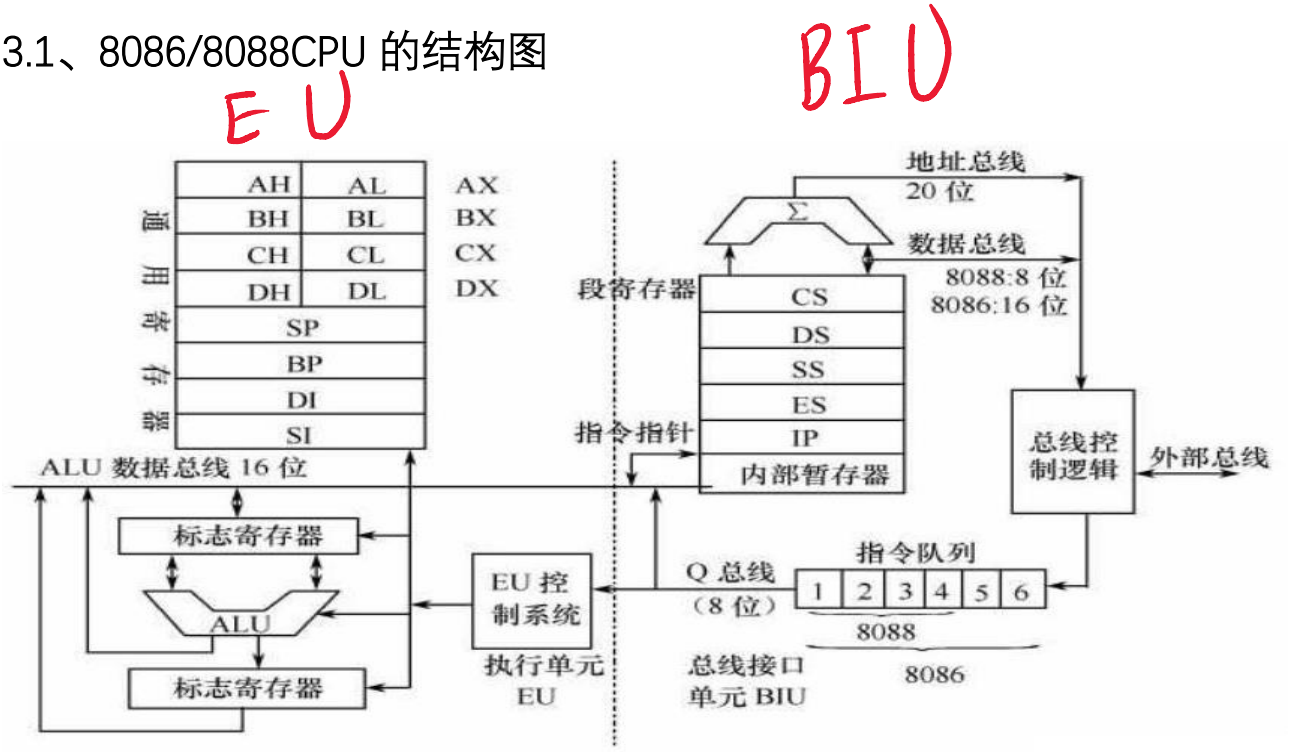
- 优点：命中率较高、速度较快、硬件较简单
- 缺点：成本高

2.7.8.4、替换策略

先进先出法（FIFO）	最先调进快存的数据块最先被替换
最近最少使用法（LRU）	最久最少使用的数据块最先被替换
随机法（RANDOM）	随意选择被替换的块，不依赖以前的使用情况

第三章 微型计算机的微处理器

3.1、8086/8088CPU 的结构图



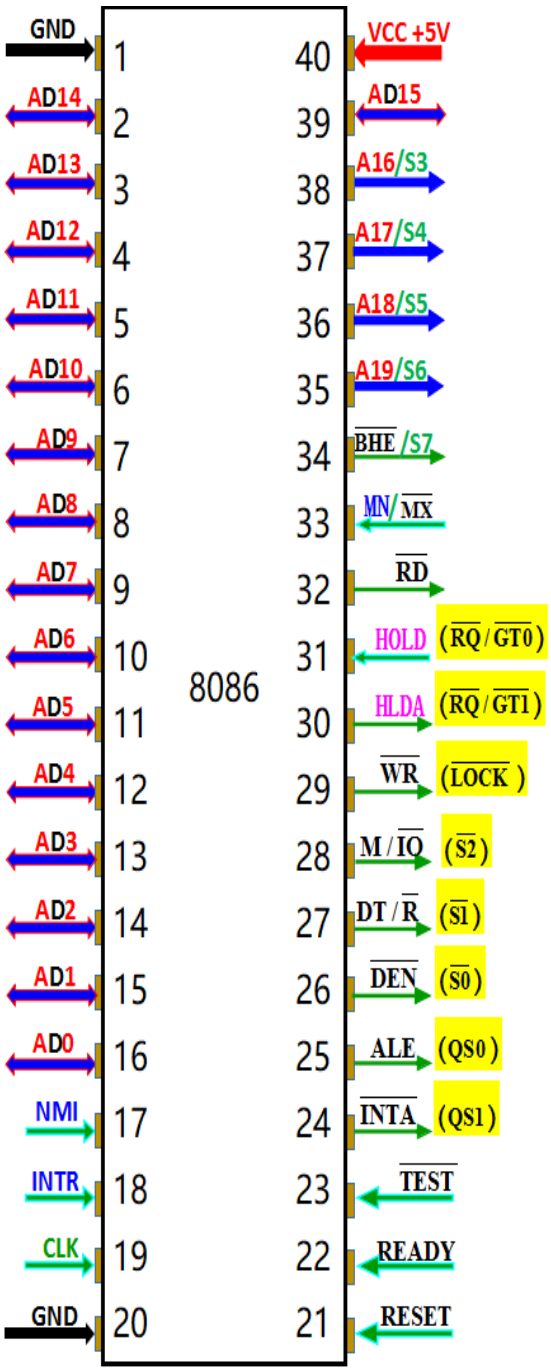
8086/8088 CPU 功能框图

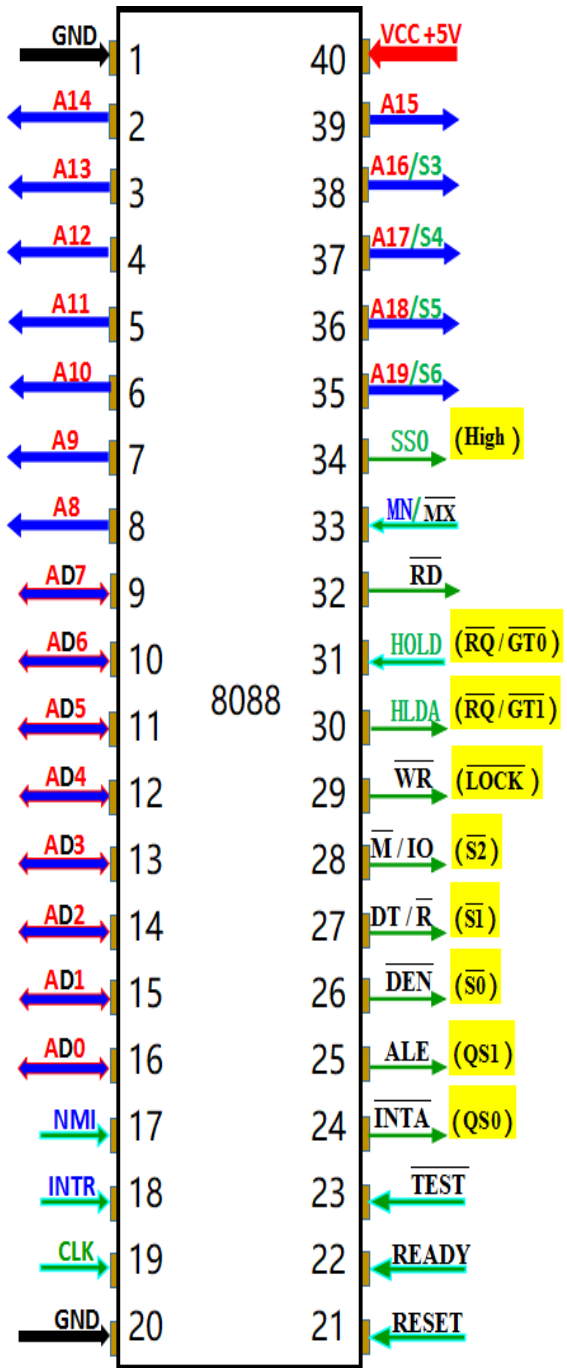
EU（执行单元）	（1）、4 个通用寄存器：AX、BX、CX、DX （2）、4 个专用寄存器：BP、SP、SI、DI （3）、标志寄存器：FR （4）、算术逻辑单元：ALU
	功能：负责执行所有的指令，向总线接口单元（BIU）提供指令执行的结果数据和地址，并对通用寄存器和标志寄存器进行管理。
BIU（总线接口单元）	（1）、4 个段寄存器：CS、DS、SS、ES （2）、指令指针寄存器：IP （3）、地址加法器 （4）、指令队列
	功能：执行外部总线周期，负责存储器与 I/O 端口传送数据，也就是负责 CPU 与存储器和外设之间的信息交换。

	数据寄存器	专用寄存器
EU 通用寄存器 (8)	AX：累加寄存器（AH、AL） BX：基址寄存器（BH、BL） CX：计数寄存器（CH、CL） DX：数据寄存器（DH、DL）	SP：堆栈指针寄存器 BP：基址指针寄存器 DI：目的变址寄存器 SI：源变址寄存器
BIU 段寄存器 (4)	CS：代码段寄存器，用于装代码段的起始地址（CS:IP） DS：数据段寄存器，用于装数据段的起始地址 SS：堆栈段寄存器，用于装堆栈段的起始地址（SS:SP） ES：附加段寄存器，用于装附加段的起始地址	
控制寄存器 (2)	IP：指令指针寄存器 FR：状态标志寄存器	

状态标志 (6)	CF	进位标志	CF=1 为加减运算时结果的最高位有进位或有借位，否则 CF=0
	PF	奇偶标志	PF=1 为运算结果的低 8 位中所含“1”的个数为偶数，否则 PF=0
	AF	辅助进位标志	AF=1 为运算结果的低 4 位向高 4 位有进位或借位，否则 AF=0
	ZF	零标志	ZF=1 为运算结果为 0，否则 ZF=0
	SF	符号标志	SF=1 为运算结果为负，否则 SF=0
	OF	溢出标志	OF=1 为运算结果溢出，否则 OF=0
控制标志 (3)	TF	单步调试	TF=0 时，无指令单步调试操作 TF=1 时，有指令单步调试操作
	IF	中断允许	IF=0 时，有可屏蔽中断请求但无中断响应产生 IF=1 时，有可屏蔽中断请求必有中断响应产生
	DF	方向标志	DF=0 时，在字符串操作时使地址指针自动增量 DF=1 时，在字符串操作时使地址指针自动减量

3.2、8086/8088CPU 的引脚图





3.2.1、数据和地址引脚

AD_{15}~AD_0	地址/数据复用引脚
1、 在T_1期间作地址线 AD_{15}~AD_0，此时输出存储单元的低 16 位地址。 2、 在T_2~T_3期间作数据线 AD_{15}~AD_0，此时是双向的。 3、 当 CPU 响应中断以及系统总线处理"保持响应"状态时， AD_{15}~AD_0被浮置为高阻状态。	

$A_{19}/S_6 \sim A_{16}/S_3$	地址/状态复用引脚
1、 在T_1期间作地址线 $AD_{19} \sim AD_{16}$，此时输出存储单元的高 4 位地址，这样就和$AD_{15} \sim AD_0$组合在一起构成了 20 位的地址总线$AD_{19} \sim AD_0$。 2、 在$T_2 \sim T_4$期间作状态线$S_6 \sim S_3$，此时S_6为低电平，表示 CPU 正与总线相连；S_5状态表示当前中断允许标志 IF 的状态，如果 IF=1 表明当前允许可屏蔽中断请求，IF=0 表明当前禁止可屏蔽中断请求；S_4和S_3状态组合起来指出 CPU 正在使用哪一个段寄存器。 3、 当 CPU 响应中断以及系统总线处理"保持响应"状态时，$AD_{19}/S_6 \sim AD_{16}/S_3$被浮置为高阻状态。	

$\overline{BHE}/S_7$	高 8 位数据总线允许/状态复用引脚
1、 在T_1期间，此引脚输出 BHE 信号，表示高 8 位数据总线$D_{15} \sim D_8$上的数据有效。 2、 在$T_2 \sim T_4$状态，此引脚输出信号S_7，不过，在当前的芯片（8086、8086-1、8086-2）设计中，S_7并未被赋予任何实际意义。	

3.2.2、读和写控制引脚

ALE	地址锁存允许信号
1、 在T_1期间，ALE 输出有效电平，以表示当前地址/数据复用总线上输出的是地址信息，地址锁存器将 ALE 作为锁存信号，对地址进行锁存。	

$\overline{WR}$	写信号
1、 有效时，表示 CPU 正在写出数据给 I/O 端口或者存储器。	

$\overline{RD}$	读信号
1、 有效时，表示 CPU 正在从 I/O 端口或者存储器读入数据。	

$\overline{DEN}$	数据允许信号
1、 有效时，表示当前数据总线上正在传送数据。	

READY	准备就绪信号
1、 有效时，表示存储器或 I/O 设备准备就绪，可以立即进行一次数据传输。 2、 CPU 在每个总线周期的T_3状态开始对 READY 信号进行检测，若检测到 READY 为高电平时，则总线周期按正常时序进行读、写操作，不需要插入等待状态T_w；若检测到 READY 为低电平时，则表示存储器或 I/O 设备暂未准备就绪，则 CPU 在T_3~T_4之间自动插入一个或几个等待状态T_w来延长总线周期，直到 CPU 检测到 READY 为高电平之后，才使 CPU 退出等待进入T_4状态，完成数据传送。	

DT/$\overline{R}$	数据发送/接收信号
1、 输出低电平时，数据由 CPU 接收。 2、 输出高电平时，数据由 CPU 发出。	

M/$\overline{IO}$	存储器/输入/输出控制信号
1、 输出低电平时，表示 CPU 将访问 I/O 端口，这时地址总线 AD_{15}~AD_0提供 16 位 I/O 端口地址。 2、 输出高电平时，表示 CPU 将访问存储器，这时地址总线 AD_{19}~AD_0提供 20 位存储器地址。	

3.2.3、中断请求和响应引脚

NMI	非屏蔽中断请求信号
1、 当该引脚电平出现由低到高变化时，不管中断允许标志 IF 是 0 还是 1，CPU 就会在当前指令周期结束后，响应中断请求，转去执行中断处理程序。	

INTR	可屏蔽中断请求信号
1、 有效时，表示外部设备有中断请求。 2、 CPU 在每一个指令周期的最后一个 T 状态检测此引脚，一旦检测到此引脚为高电平并且中断允许标志位 IF=1，则 CPU 在当前指令周期结束后，响应中断请求，转去执行中断处理程序。	

$\overline{INTA}$	中断响应信号
1、 $\overline{INTA}$是 CPU 响应可屏蔽中断后发给请求中断设备的回答信号，对中断请求信号INTR的响应，目的是为了获取中断类型码。	

3.2.4、总线请求和响应引脚

HOLD	**总线请求信号**
1、 在最小模式系统中，除 CPU（8086/8088）以外的其它总线控制器，如 DMA 控制器申请使用系统总线的请求信号。	

HLDA	**总线请求响应信号**
1、 有效时，表示 CPU 对其它主部件的总线请求作出响应，与此同时，所有与三态门相接的 CPU 的引脚呈现高阻抗，从而让出了总线。	

3.2.5、其它的特殊引脚

RESET	**复位信号引脚**
1、 有效时，将使 CPU 回到其初始状态，CPU 内部的寄存器初始化，CS=FFFFH、IP=0000H。	

$\overline{\text{TEST}}$	**测试信号引脚**
1、 $\overline{\text{TEST}}$信号和 WAIT 指令配合使用，当 CPU 执行 WAIT 指令时，CPU 处于等待状态，一旦检测到$\overline{\text{TEST}}$信号为低电平时，则结束等待状态，CPU 继续往下执行被暂停的指令。	

MN/$\overline{\text{MX}}$	**组态选择**
1、 接高电平，工作在最小组态。 2、 接低电平，工作在最大组态。	

3.3、8086/8088CPU 的工作模式

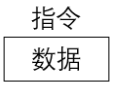
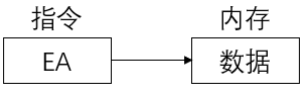
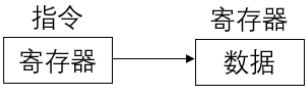
最小模式：在系统中只有 8086/8088 一个微处理器，在该系统中，所有的总线控制信号都直接由 8086/8088 产生，因此，系统中总线控制电路被减到最少。

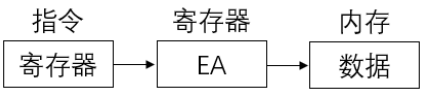
最大模式：在系统中包含两个或者多个微处理器，其中一个主处理器就是 8086/8088，其它处理器为协处理器，是协助主处理器工作的。它作用在中等规模或大型的 8086/8088 系

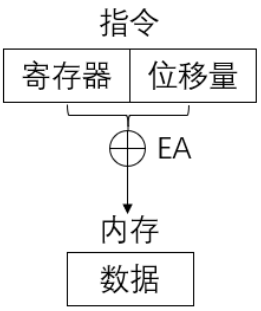
统中。一般情况下和 8086/8088 配合的协处理器有两个，一个是数值运算协处理器 8087，一个是输入/输出协处理器 8089。

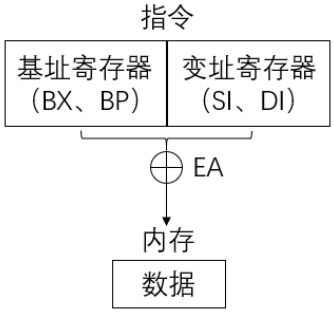
第四章 微型计算机的指令系统

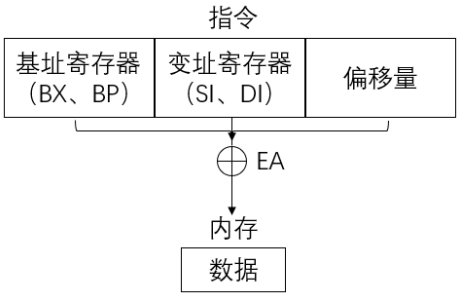
4.1、8086/8088CPU 的寻址方式

立即数寻址	MOV AL,80H MOV AX,1090H	
直接寻址	MOV AL,[2000H] MOV AX,[2000H]	
寄存器寻址	MOV AL,BL MOV AX,BX	

寄存器间接寻址	**数据段基址寻址（DS）：**
	MOV AX,[BX]
	MOV AX,[SI]
	MOV AX,[DI]
	堆栈段基址寻址（SS）：
	MOV AX,[BP]
		

寄存器相对寻址	**数据段基址寻址（DS）：** MOV AX,[BX+8/16 位偏移量] MOV AX,[SI+8/16 位偏移量] MOV AX,[DI+8/16 位偏移量] **堆栈段基址寻址（SS）：** MOV AX,[BP+8/16 位偏移量]	

基址加变址寻址	**数据段基址寻址（DS）：**
	MOV AX,[BX+SI]
	MOV AX,[BX+DI]
	MOV AX,[BP+SI]
	堆栈段基址寻址（SS）：
	MOV AX,[BP+DI]
		

相对基址加变址寻址	**数据段基址寻址（DS）：** MOV AX,[BX+SI+8/16 位偏移量] MOV AX,[BX+DI+8/16 位偏移量] MOV AX,[BP+SI+8/16 位偏移量] **堆栈段基址寻址（SS）：** MOV AX,[BP+DI+8/16 位偏移量]	

4.2、8086/8088CPU 的指令分类

4.2.1、传送类指令

指令类型	指令格式	指令功能	状态标志位						指令备注	
			O	S	Z	A	P	C		
通用数据传送	MOV 目标，源	传送字节或字	-	-	-	-	-	-	源：寄存器、存储器、立即数	目标：寄存器、存储器
	XCHG 目标，源	交换字节或字	•	•	•	•	•	•	源：通用寄存器、存储器	目标：通用寄存器、存储器
	PUSH 源	字压入堆栈	-	-	-	-	-	-	源：寄存器、存储器（SP=SP-2）	
	POP 目标	字弹出堆栈	-	-	-	-	-	-	目标：寄存器（CS除外）、存储器（SP=SP+2）	
	XLAT	字节翻译	-	-	-	-	-	-		
目标地址传送	LEA 目标，源	装入有效地址	-	-	-	-	-	-	源：内存操作数	目标：16位通用寄存器
	LDS 目标，源	装入数据段指针到DS	-	-	-	-	-	-	源：内存操作数	目标：16位通用寄存器
	LES 目标，源	装入附加段指针到ES	-	-	-	-	-	-	源：内存操作数	目标：16位通用寄存器
I/O数据传送	IN 累加器，端口	输入字节或字	-	-	-	-	-	-	累加器：AL或AX	端口：地址0~255或间址寄存器DX
	OUT 端口，累加器	输出字节或字	-	-	-	-	-	-	累加器：AL或AX	端口：地址0~255或间址寄存器DX
说明：•运算结果会影响标志位、-运算结果不影响标志位、*运算结果表示为任意值										

4.2.2、操作类指令

4.2.2.1、算术运算类指令

指令类型	指令格式	指令功能	状态标志位						指令备注	
			O	S	Z	A	P	C		
加法指令	ADD 目标，源	加法（字节/字）	•	•	•	•	•	•	源：通用寄存器、存储器、立即数	目标：通用寄存器、存储器
	ADC 目标，源	带进位加法（字节/字）	•	•	•	•	•	•	源：通用寄存器、存储器、立即数	目标：通用寄存器、存储器
	INC 目标	加1（字节/字）	•	•	•	•	•	–		目标：通用寄存器、存储器
减法指令	SUB 目标，源	减法（字节/字）	•	•	•	•	•	•	源：通用寄存器、存储器、立即数	目标：通用寄存器、存储器
	SBB 目标，源	带借位减法（字节/字）	•	•	•	•	•	•	源：通用寄存器、存储器、立即数	目标：通用寄存器、存储器
	DEC 目标	减1（字节/字）	•	•	•	•	•	–		目标：通用寄存器、存储器
	NEG 目标	取补	•	•	•	•	•	1		目标：通用寄存器、存储器
	CMP 目标，源	比较	•	•	•	•	•	•	源：通用寄存器、存储器、立即数	目标：通用寄存器、存储器
乘法指令	MUL 源	无符号乘法（字节/字）	•	*	*	*	*	•	源：通用寄存器、存储器（8位:AL×源=AH+AL、16位:AX×源=DX+AX）	
	IMUL 源	有符号乘法（字节/字）	•	*	*	*	*	•	源：通用寄存器、存储器（8位:AL×源=AH+AL、16位:AX×源=DX+AX）	
除法指令	DIV 源	无符号除法（字节/字）	*	*	*	*	*	*	源：通用寄存器、存储器（8位:AH+AL/源=AL·AH、16位:DX+AX/源=AX·DX）	
	IDIV 源	有符号除法（字节/字）	*	*	*	*	*	*	源：通用寄存器、存储器（8位:AH+AL/源=AL·AH、16位:DX+AX/源=AX·DX）	
	CBW	字节转换为字	–	–	–	–	–	–		
	CWD	字转换为双字	–	–	–	–	–	–		
十进制调整	DAA	加法的十进制调整	*	•	•	•	•	•		
	DAS	减法的十进制调整	•	•	•	•	•	•		
	AAA	加法的ASCII码调整	*	*	*	•	*	1		
	AAS	减法的ASCII码调整	*	*	*	•	*	•		
	AAM	乘法的ASCII码调整	*	•	•	*	•	*		
	AAD	除法的ASCII码调整	*	•	•	–	•	*		
说明：•运算结果会影响标志位、–运算结果不影响标志位、*运算结果表示为任意值										

4.2.2.2、逻辑运算类指令

指令类型	指令格式	指令功能	状态标志位						指令备注	
			O	S	Z	A	P	C		
逻辑运算	NOT 目标	非（字节/字）	–	–	–	–	–	–		目标：通用寄存器、存储器
	AND 目标，源	与（字节/字）	0	•	•	*	•	0	源：通用寄存器、存储器、立即数	目标：通用寄存器、存储器
	OR 目标，源	或（字节/字）	0	•	•	*	•	0	源：通用寄存器、存储器、立即数	目标：通用寄存器、存储器
	XOR 目标，源	异或（字节/字）	0	•	•	*	•	0	源：通用寄存器、存储器、立即数	目标：通用寄存器、存储器
	TEST 目标，源	测试（字节/字）	0	•	•	*	•	0	源：8位或16位立即数	目标：通用寄存器、存储器
说明：•运算结果会影响标志位、–运算结果不影响标志位、*运算结果表示为任意值										

4.2.2.3、移位运算类指令

指令类型	指令格式	指令功能	状态标志位						指令备注	
			O	S	Z	A	P	C		
移位运算	SHL 目标，计数值	逻辑左移（字节/字）	•	•	•	*	•	•	目标：通用寄存器、存储器	计数值：1或CL（移位次数）
	SHR 目标，计数值	逻辑右移（字节/字）	•	•	•	*	•	•	目标：通用寄存器、存储器	计数值：1或CL（移位次数）
	SAL 目标，计数值	算术左移（字节/字）	•	•	•	*	•	•	目标：通用寄存器、存储器	计数值：1或CL（移位次数）
	SAR 目标，计数值	算术右移（字节/字）	•	•	•	*	•	•	目标：通用寄存器、存储器	计数值：1或CL（移位次数）
循环移位	ROL 目标，计数值	循环左移（字节/字）	•	–	–	*	–	•	目标：通用寄存器、存储器	计数值：1或CL（移位次数）
	ROR 目标，计数值	循环右移（字节/字）	•	–	–	*	–	•	目标：通用寄存器、存储器	计数值：1或CL（移位次数）
	RCL 目标，计数值	带进位循环左移（字节/字）	•	–	–	*	–	•	目标：通用寄存器、存储器	计数值：1或CL（移位次数）
	RCR 目标，计数值	带进位循环右移（字节/字）	•	–	–	*	–	•	目标：通用寄存器、存储器	计数值：1或CL（移位次数）
说明：•运算结果会影响标志位、–运算结果不影响标志位、*运算结果表示为任意值										

4.2.3、串操作指令

指令类型	指令格式	指令功能	状态标志位						指令备注
			O	S	Z	A	P	C	
串基本操作	MOVSB	字节串传送	–	–	–	–	–	–	可添加串重复前缀：REP
	MOVSW	字串传送	–	–	–	–	–	–	可添加串重复前缀：REP
	CMPSB	字节串比较	•	•	•	•	•	•	可添加串重复前缀：REPE/REPZ 和 REPNE/REPNZ
	CMPSW	字串比较	•	•	•	•	•	•	可添加串重复前缀：REPE/REPZ 和 REPNE/REPNZ
	SCASB	字节串扫描	•	•	•	•	•	•	可添加串重复前缀：REPE/REPZ 和 REPNE/REPNZ
	SCASW	字串扫描	•	•	•	•	•	•	可添加串重复前缀：REPE/REPZ 和 REPNE/REPNZ
	LODSB	读字节串	–	–	–	–	–	–	可添加串重复前缀：无
	LODSW	读字串	–	–	–	–	–	–	可添加串重复前缀：无
	STOSB	写字节串	–	–	–	–	–	–	可添加串重复前缀：REP
	STOSW	写字串	–	–	–	–	–	–	可添加串重复前缀：REP
串重复前缀	REP	无条件重复	–	–	–	–	–	–	重复前提：（1）、CX≠0，表示重复次数还未满 （2）、ZF=1 ，表示目的操作数等于（源操作数/扫描值）
	REPE/REPZ	当（相等/为零）时重复	–	–	–	–	–	–	
	REPNE/REPNZ	当（不相等/不为零）时重	–	–	–	–	–	–	
说明：•运算结果会影响标志位、–运算结果不影响标志位、*运算结果表示为任意值									

隐含参数	对应单元
源字符串的起始地址	DS:SI
目标字符串的起始地址	ES:DI
重复次数	CX
SCAS指令的扫描值	AL/AX
LODS指令的目的操作数	AL/AX
STOS指令的源操作数	AL/AX
传送方向	DF=0，SI、DI自动增量（注意：MOVS指令中，源字符串地址高于目标字符串地址）
	DF=1、SI、DI自动减量（注意：MOVS指令中，源字符串地址低于目标字符串地址）

4.2.4、控制类指令

4.2.4.1、处理机控制指令

指令类型	指令格式	指令功能	影响标志位								
			O	D	I	T	S	Z	A	P	C
对标志位操作	CMC	取反进位标志	–	–	–	–	–	–	–	–	C
	CLC	清除进位标志	–	–	–	–	–	–	–	–	0
	STC	置1进位标志	–	–	–	–	–	–	–	–	1
	CLD	清除方向标志	–	0	–	–	–	–	–	–	–
	STD	置1方向标志	–	1	–	–	–	–	–	–	–
	CLI	清除中断标志	–	–	0	–	–	–	–	–	–
	STI	置1中断标志	–	–	1	–	–	–	–	–	–
外部同步操作	ESC	交权	–	–	–	–	–	–	–	–	–
	WAIT	等待	–	–	–	–	–	–	–	–	–
	LOCK	封锁总线	–	–	–	–	–	–	–	–	–
其它常见操作	HLT	暂停	–	–	–	–	–	–	–	–	–
	NOP	空操作	–	–	–	–	–	–	–	–	–
说明：•运算结果会影响标志位、–运算结果不影响标志位、*运算结果表示为任意值											

4.2.4.2、程序转移类指令

指令类型		指令格式	指令功能	测试条件
无条件转移		JMP 目标标号	无条件转移	
		CALL 过程名称	过程调用	
		RET 弹出值	过程返回	
有条件转移	对无符号数	JA/JNBE 目标标号	（大于/不小于也不等于）	CF=0且ZF=0
		JB/JNAE 目标标号	（小于/不大于也不等于）	CF=1且ZF=0
		JAE/JNB 目标标号	（大于或等于/不小于）	CF=0或ZF=1
		JBE/JNA 目标标号	（小于或等于/不大于）	CF=1或ZF=1
	对有符号数	JG/JNLE 目标标号	（大于/不小于也不等于）	OF⊕SF=0且ZF=0
		JL/JNGE 目标标号	（小于/不大于也不等于）	OF⊕SF=1且ZF=0
		JGE/JNL 目标标号	（大于或等于/不小于）	OF⊕SF=0或ZF=1
		JLE/JNG 目标标号	（小于或等于/不大于）	OF⊕SF=1或ZF=1
	位条件转移	JE 目标标号	（等于）	ZF=1
		JNE 目标标号	（不等于）	ZF=0
		JZ 目标标号	零标志位为1	ZF=1
		JNZ 目标标号	零标志位为0	ZF=0
		JC 目标标号	进位标志位为1	CF=1
		JNC 目标标号	进位标志位为0	CF=0
		JO 目标标号	溢出标志位为1	OF=1
		JNO 目标标号	溢出标志位为0	OF=0
		JP 目标标号	奇偶标志位为1	PF=1
		JNP 目标标号	奇偶标志位为0	PF=0
		JS 目标标号	符号标志位为1	SF=1
		JNS 目标标号	符号标志位为0	SF=0
循环控制		LOOP 目标标号	（循环指令）	CX≠0
		LOOPE/LOOPZ 目标标号	（等于/结果为零）	CX≠0且ZF=1
		LOOPNE/LOOPNZ 目标标号	（不等于/结果不为零）	CX≠0且ZF=0
		JCXZ 目标标号	（CX内容为零则转移）	
中断控制		INT 中断类型	中断	
		INTO	溢出中断	
		IRET	中断返回	

第五章 微型计算机的汇编程序

5.1、汇编语言的基础知识

5.1.1、伪指令语句格式

[标号名] 伪指令 操作数 [注释]

5.1.2、常数、变量和标号

常数	**概述：常数是没有任何属性的纯数值** **注意：常数或者常量是没有地址的** （1）、二进制　　：以字母 B 结尾　　，例如：01011010B （2）、八进制　　：以字母 O 或 Q 结尾　　，例如：723O、65Q （3）、十进制　　：以字母 D 或无结尾　　，例如：129D、129 （4）、十六进制　：以字母 H 结尾　　，例如：0FF00H、80H （5）、实数　　：±整数部分.小数部分 E±指数部分，例如：5.123E-4 （6）、字符串常数：使用引号引起来一个或多个字符 ，例如：‘AB’、”ABC”
变量	**概述：变量代表存放在某些存储单元的数据，其值在程序运行期间可修改。** **属性：段属性、偏移属性、类属性** （1）、数据定义伪指令 DB（字节型）、DW（字型）、DD（双字型）、… （2）、第一种定义： 变量名　数据定义伪指令　表达式 1，表达式 2,… （3）、第二种定义： 变量名　数据定义伪指令　表达式 1　DUP(表达式 2) （4）、变量初始化： VAR_DATA SEGMENT DATA1 DB 12H,12H,12H,12H DATA2 DB 10H DUP(4) VAR_DATA ENDS
标号	**概述：标号是一条指令目标代码的符号地址，它常作为转移指令的操作数。** **属性：段属性、偏移属性、类属性**

5.1.3、表达式与运算符

表达式由操作数和运算符组成，在汇编时一个表达式的到一个值。

5.1.3.1、算术运算符

+	加法操作符	例如：1H + 1H（=2H）
-	减法操作符	例如：2H – 1H（=1H）
*****	乘法操作符	例如：1H * 3H（=3H）
/	除法操作符	例如：6H / 3H（=2H）
MOD	取余操作符	例如：15H MOD 2（=01H）
SHL	左移操作符	例如：21H SHL　2（=84H）
SHR	右移操作符	例如：10H SHR　2（=04H）

5.1.3.2、逻辑运算符

AND	逻辑“与”操作符	例如：24H AND 0FH（=04H）
OR	逻辑“或”操作符	例如：24H OR　0FH（=2FH）
XOR	逻辑“异或”操作符	例如：24H XOR 0FH（=2BH）
NOT	逻辑“非”操作符	例如：NOT 0FH（=0DBH）

5.1.3.3、关系运算符

EQ	等于	例如：25 EQ 25（=0FFFFH）
NE	不等于	例如：25 NE 23（=0FFFFH）
LT	小于	例如：25 LT 26（=0FFFFH）
LE	小于等于	例如：25 LE 25（=0FFFFH）
GT	大于	例如：26 GT 25（=0FFFFH）
GE	大于等于	例如：24 GE 24（=0FFFFH）

注意事项:

（1）、当为真取（0FFFFH），当为假取（0000H）。

5.1.3.4、分析运算符

SEG	求段基址	格式：SEG　　符号名
OFFSET	求偏移地址	格式：OFFSET　符号名
TYPE	求符号名类型	格式：TYPE　　符号名
SIZE	求为符号名分配的字节数	格式：SIZE　　符号名
LENGTH	求为符号名分配的项数	格式：LENGTH　符号名

注意事项:

(1)、SIZE 和 LENGTH 所求的数据项必须是用重复格式 DUP()定义的。

(2)、(SIZE　符号名）=（LENGTH　符号名）＊（TYPE　符号名）

(3)、符号名类型值表:

类型	**1 字节 BYTE**	**2 字节 WORD**	**4 字节 DWORD**	**8 字节 QWORD**	**10 字节 TBYTE**	**近程 NEAR**	**远程 FAR**
类型值	**1**	**2**	**4**	**8**	**10**	**-1**	**-2**

5.1.3.5、组合运算符

PTR	为符号名建立新类型	格式：符号名新类型 PTR 符号名
THIS	为符号名建立新类型	格式：THIS 符号名新类型

5.1.3.6、分离运算符

LOW	取低字节	格式：LOW　表达式
HIGH	取高字节	格式：HIGH 表达式

5.1.3.7、运算优先级

优先级		运算符
高级	0	括号中表达式
	1	LENGTH、SIZE、WIDTH、MASK
	2	PTR、OFFSET、SEG、TYPE、THIS、段前缀
	3	HIGH、LOW
	4	*、/、MOD、SHL、SHR
	5	+、-
	6	EQ、NE、LT、LE、GT、GE
	7	NOT
	8	AND
	9	OR、XOR
低级	10	SHORT

5.2、汇编语言的伪指令

5.2.1、符号定义伪指令

用 EQU 语句赋值的符号名不能被重新赋值	格式：名字 EQU 表达式
用“ = ”语句赋值的符号名允许被重新赋值	格式：名字 = 表达式

5.2.2、数据定义伪指令

1 字节定义伪指令（字节）	格式：名字 DB 表达式或数据项表
2 字节定义伪指令（字）	格式：名字 DW 表达式或数据项表
4 字节定义伪指令（双字）	格式：名字 DD 表达式或数据项表
8 字节定义伪指令（四字）	格式：名字 DQ 表达式或数据项表
10 字节定义伪指令	格式：名字 DT 表达式或数据项表

5.2.3、过程定义伪指令

过程名 PROC [NEAR/FAR] 过程名 ENDP

5.2.4、段定义伪指令

段名称 SEGMENT [定位方式][连接方式]['类别名'] 段名称 ENDS

5.2.5、段说明伪指令

ASSUME CS:CODE,DS:DATA

5.2.6、定位伪指令

从表达式指定的偏移地址开始连续存放	格式：ORG 表达式

5.3、汇编语言的系统功能调用

功能名	功能号	出口参数	示例代码
输入单字符	**01H**	输入字符存入 AL	`MOV AH,1` `INT 21H`

输出单字符	**02H**	输出 DL 中的字符	`MOV DL,'A'`
			`MOV AH,2`
			`INT 21H`

输出字符串	**09H**	输出缓冲区的字符串	… BUF DB 'hello,world','$' … MOV DX,OFFSET BUF MOV AH,9 INT 21H

输入字符串	**0AH**	输入字符串到缓冲区	… BUF DB 20 ;缓冲区大小 DB ? ;实际字符数 DB 20 DUP(?) ;输入字符串 … MOV DX,OFFSET BUF MOV AH,0AH INT 21H
结束程序	**4CH**	结束当前程序并返回系统	MOV AH,4CH INT 21H

5.4、汇编语言的三种基本结构

(1)、顺序结构

(2)、分支结构

(3)、循环结构

5.5、汇编语言的程序框架

```
DATA SEGMENT

DATA ENDS

CODE SEGMENT
      ASSUME CS:CODE,DS:DATA
START:
      MOV AX,DATA
      MOV DS,AX

      MOV AH,4CH
      INT 21H
CODE ENDS
      END START
```

5.6、汇编语言的常见使用

判断奇偶：TEST AL,01H
- ZF=0，奇数，JNZ
- ZF=1，偶数，JZ

判断正负：TEST AL,80H
- ZF=0，负数，JNZ、JS
- ZF=1，正数，JZ、JNS

判正负零：AND AX,AX
- JZ 标识，若 ZF=1，表示 AX 为零，跳转到标识
- JS 标识，若 SF=1，表示 AX 为负，跳转到标识
- 以上两种都不符合即为正数，直接进行操作即可

大写变小写：
- ADD AL,20H
- OR AL,20H

小写变大写：SUB AL,20H

将寄存器清零：
- MOV AX,0
- OR AX,0
- SUB AX,AX
- XOR AX,AX

第六章 微型计算机的中断技术

6.1、中断的概述

中断是指计算机由任何非寻常的、非预期的、急需处理的事件引起 CPU 暂时中断现行程序的执行，而转去执行另一服务程序来处理这些事情，等处理完后又返回原程序，我们把这一执行过程叫做中断。

6.2、中断的分类

(1)、外部中断：包括可屏蔽中断请求和非屏蔽中断请求

(2)、内部中断：包括 CPU 产生的中断（除法、单步），软件中断（中断指令产生的中断）

6.3、中断的中断源

(1)、外部设备，例如：打印机、键盘等

(2)、软件调用，例如：系统功能调用、设置断点运行程序等

(3)、数据源，例如：磁盘、磁带等

(4)、故障源，例如：电源掉电、除法运算出错等

(5)、实时时钟

6.4、中断的优先级

一般一个系统中会有多个中断源，当某时刻出现两个或多个中断源申请中断请求时，中断系统就需要判别中断源的优先级，并按优先级的高低决定响应的顺序。

除法中断 > INT N > 非屏蔽中断(NMI) > 可屏蔽中断(INTR) > 单步中断

6.5、中断的类型码

中断类型码	中断类型
0	除法错误中断
1	单步中断
2	非屏蔽中断
3	断点中断
4	溢出中断
5	越界中断
6	非法操作码中断
7	浮点单元不可用中断
8	双重故障中断
9	保留
10	无效任务状态段中断
11	段不存在中断
12	堆栈异常中断
13	一般保护中断
14	页故障中断
15	保留
16	浮点错误中断
17	对准检查中断
18–31	保留
32–255	INT N指令中断和INTR可屏蔽中断

6.6、中断向量表

中断向量是中断处理子程序的入口地址，每个中断类型码对应一个中断向量。

中断向量表是指中断类型码与中断向量的关系。

中断向量表位于存储器的最低部地址为 000H~3FFH，共 1K 字节单元。

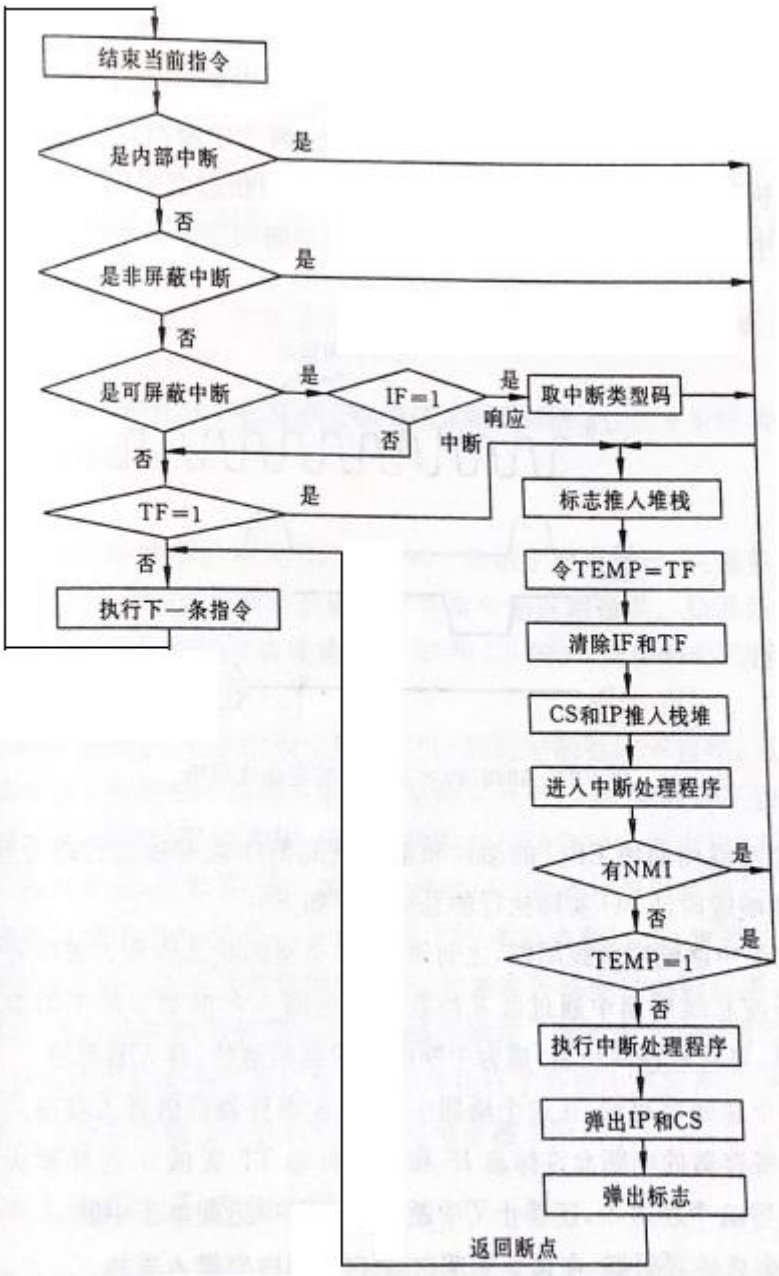
6.7、中断系统的功能

(1)、实现中断及返回

(2)、实现优先权排队

(3)、实现高级中断源能中断低级的中断处理

6.8、中断响应及处理过程



8086/8088 的中断响应流程图

第七章 微型计算机的接口技术

7.1、接口的概述

输入输出(I/O)是指微型计算机与外界的信息交换，即通信。微型计算机与外界的通信，

是通过输入输出设备进行的，通常一种 I/O 设备与微型机连接，就需要一个连接电路，我们称之为 I/O 接口。

接口是用于控制微机系统与外设或外设与系统设备之间的数据交换和通信的硬件电路。接口设计涉及到两个基本问题，一是中央处理器如何寻址外部设备，实现多个设备的识别；二是中央处理器如何与外设连接，进行数据、状态和控制信号的交换。

7.2、接口的功能

(1)、设置数据的寄存、缓冲逻辑，以适应 CPU 与外设之间的速度差异。

(2)、进行信息格式的转换，如串行和并行的转换。

(3)、协调 CPU 与外设在信息类型和电平上的差异，如电平转换驱动器、数/模和模/数转换器等。

(4)、协调时序差异，同步 CPU 与外设的工作。

(5)、地址译码和设备选择功能，使 CPU 在某一时刻只能选中一个 I/O 端口。

(6)、提供联络信号，承担 CPU 与外设之间的联络工作，联络的具体信息有控制信息、状态信息和请求信号等，如外设的“Ready”、“Busy”等状态。

(7)、设置中断和 DMA 控制逻辑，以保证在中断和 DMA 允许的情况下，产生中断和 DMA 请求信号，并在接受到中断和 DMA 应答之后完成中断处理和 DMA 传输。

7.3、接口的编址方式

独立编址方式	存储器地址和 I/O 地址分开，CPU 具有专用的 I/O 指令，系统总线具有区别存储器读写和 I/O 操作的控制信号，并以此区别地址总线上的地址是存储器地址还是 I/O 地址
统一编址方式	存储器地址和 I/O 统一考虑，地址空间的一部分是存储器，另一部分是 I/O，支持存储器操作的指令都可用于 I/O 操作

7.4、接口的数据传送方式

并行通信	并行通信是把一个字符的各数位用几条线同时进行传输，传输速度快，信息率高，但所用电缆比串行通信要多，因此，并行通信常用在传输距离较短（几米~几十米）和数据传输率较高的场合
串行通信	串行通信是把数据一位一位的依次传输，每一位数据占据一个固定的时间长度，传输速度慢。但所用电缆比并行通信要少，因此，串行通信特别适合于计算机与计算机、计算机与外部设备之间的远距离通信

7.5、接口的数据控制方式

程序控制方式	**无条件传送方式**	常用于简单设备，CPU 认为它们总是处于就绪状态，随时进行数据传送，适合简单、慢速的数据传输。
	查询式传送方式	CPU 首先查询外设工作状态，在外设就绪时进行数据传送，效率低，适合慢速的数据传输。
中断控制方式		在外设就绪时，通过请求引脚信号，主动向 CPU 提出交换数据的请求，CPU 无其它更紧迫任务，则执行中断服务程序完成一次数据传送，适合少量、高速的数据传输。
DMA 控制方式		外设和内存间直接建立传输通道，传输过程由 DMAC 控制，传输过程可与 CPU 执行任务并行，效率高，适合大量、高速的数据传输。

7.6、并行接口和串行接口

并行接口	**串行接口**
实现并行通信的接口就是并行接口	实现串行通信的接口就是串行接口

7.7、三态门接口和锁存器接口

三态门接口	锁存器接口
（1）、三态门是一种重要的总线接口电路 （2）、三态门不具备数据保存功能，一般只用作数据输入接口，不能直接用于数据输出接口 （3）、三态门其输出可以是一般二值逻辑电路，即正常的高电平（1）或低电平（0），又可以保持特有的高阻抗状态	（1）、输出接口通常采用具有信息存储能力的器件来实现，例如：D 触发器、8 路锁存器

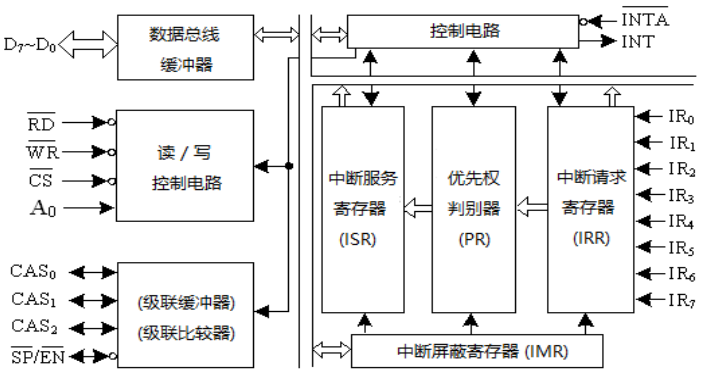
第八章 常见的三大芯片

8.1、可编程中断控制器 8259A

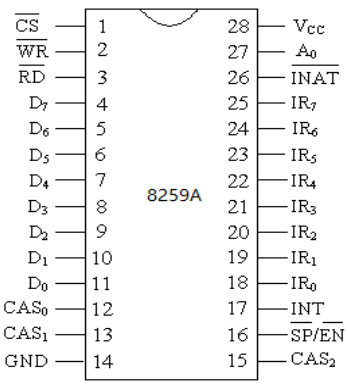
8.1.1、概述

可编程中断控制器 8259A 可对外部多个中断源进行管理，它具有 8 级优先权控制，通过级联可扩展至 64 级优先权控制。每一级中断都可以屏蔽或允许。在中断响应周期，8259A 可提供相应的中断向量，从而能迅速地转至中断服务程序。

8.1.2、内部结构



8259A 内部结构流程图



8259A 芯片引脚信号

数据总线缓冲器	这是一个双向 8 位 3 态缓冲器，是 8259A 与 CPU 交换数据的必经之路。
读 / 写控制电路	用来接收来自 CPU 的读/写控制命令和片选控制信息。 当 CPU 执行 OUT 指令，$\overline{WR}$和A_0配合，将数据总线(D_7~D_0)送来的控制字写入 8259A 当 CPU 执行 IN 指令，$\overline{RD}$和A_0配合，将 8259A 内容通过数据总线(D_7~D_0)传送给 CPU
级联缓冲器级联比较器	一片 8259A 只能接收 8 级中断从IR_7~IR_0输入，当中断超过 8 级时，可用多片 8259A 级联使用，构成主从关系。 (1)、当 8259A 处于缓冲状态时： $\overline{EN}=0$，表示 CPU 被 8259A 输入数据 $\overline{EN}=1$，表示 CPU 向 8259A 输出数据 (2)、当 8259A 处于非缓冲状态： $\overline{SP}=0$，表示从 8259A，CAS_2~CAS_0是输入信号 $\overline{SP}=1$，表示主 8259A，CAS_2~CAS_0是输出信号
中断请求寄存器 IRR	用来存放外部输入的中断请求信号。
优先权判别器 PR	用来识别各种中断信号的优先级别。 一般处理原则是：高级中断可以打断低级中断，但是低级中断不可以打断高级中断，也不允许同级中断之间互相打断。
中断服务寄存器 ISR	用来记录正在处理中的中断请求。 当任何一级中断被响应，CPU 转去执行它的中断服务程序时，ISR 寄存器中相应位置“1”，一直保持到该级中断处理过程结束为止。多重中断情况下，ISR 寄存器中可有多位被同时置“1”。
中断屏蔽寄存器 IMR	用来存放对各级中断请求的屏蔽信息。 当该寄存器中某一位置“1”时，表示禁止这一级中断请求进入系统，通过 IMR 寄存器可实现对各级中断的有选择屏蔽。
控制电路	它是 8259A 内部的控制器，8259A 芯片在控制电路控制之下构成了一个有机整体。

8.1.3、工作方式

8.1.3.1、工作方式 0-完全嵌套方式

ISR 寄存器中某位置“1”，表示 CPU 正在处理这一级中断请求，8259A 允许比它级别高的中断请求进入，禁止与它同级或低级中断请求进入。

IR_i引入的中断请求有固定的中断级别， IR_0最低，IR_7最高。

8259A 在完全嵌套方式下，可采用以下 3 种中断结束方式：

①、普通 EOI 方式：当任何一级中断服务程序结束时，只给 8259A 传送一个 EOI 结束命令，8259A 收到 EOI 命令后，自动将 ISR 寄存器中级别最高的置“ 1 ”位清“ 0 ”。

②、特殊 EOI 方式：在普通 EOI 方式的基础上，当中断服务程序结束给 8259A 发出 EOI 命令的同时，将当前结束的中断级别也传送给 8259A，就被称作特殊 EOI 方式。

③、自动 EOI 方式：当任何一级中断被响应后，ISR 寄存器中相应位置“1”，CPU 将进入中断响应总线周期，在第 2 个中断响应信号（$\overline{INTA}$）结束时，自动将 ISR 寄存器中相应位清“0”，被称作自动 EOI 方式。

8.1.3.2、工作方式 1-自动循环方式

IR_n~IR_0 优先级别不固定。每当任何一级中断被处理完，它的优先级别就被改变为最低，而将最高级赋给比它低一级的中断请求。

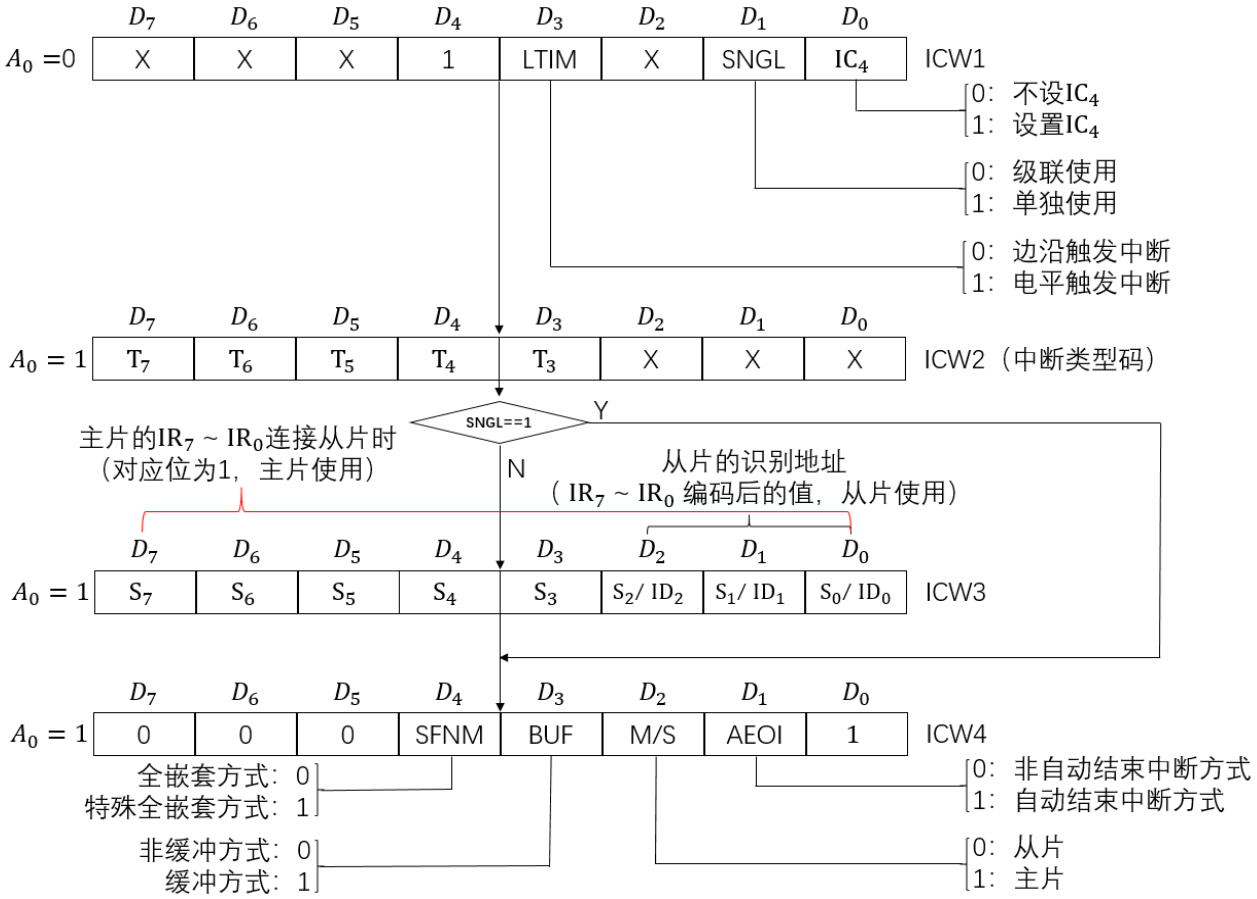
8.1.3.3、工作方式 2-中断屏蔽方式

(1)、普通屏蔽方式：将 IMR 中的某一位或某几位置“1”，可将相应级的中断请求屏蔽掉。

(2)、特殊屏蔽方式：当 CPU 正在处理某级中断时，要求仅对本级中断进行屏蔽，而允许其它优先级比它高或低的中断进入系统。

8.1.4、控制字

8.1.4.1、初始化控制字



8.1.4.2、操作控制字

8.1.5、初始化程序

单独使用	级联使用

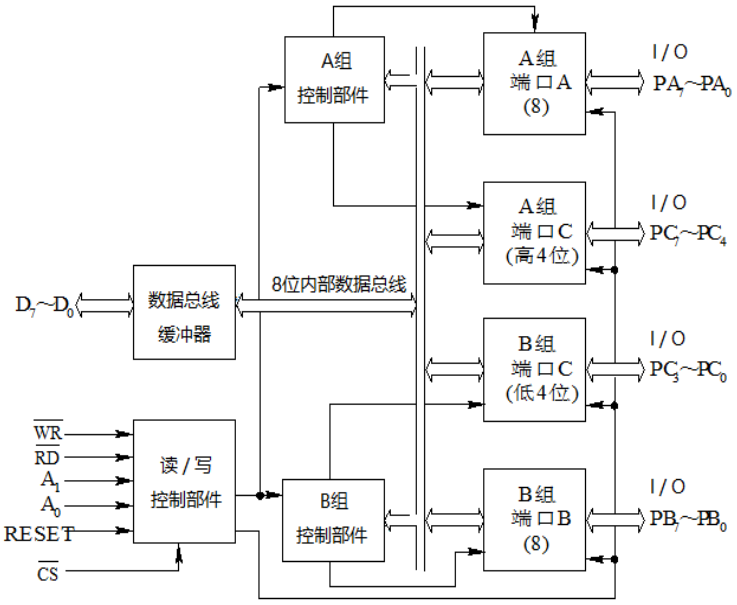
;主片初始化
MOV DX,主片端口 0
MOV AL,ICW1
OUT DX,AL
 	
MOV DX,主片端口 1
MOV AL,ICW2
OUT DX,AL
[MOV AL,ICW3]
[OUT DX,AL]
MOV AL,ICW4
OUT DX,AL
	;主片初始化
	MOV DX,主片端口 0
	MOV AL,ICW1
	OUT DX,AL
	
	MOV DX,主片端口 1
	MOV AL,ICW2
	OUT DX,AL
	MOV AL,ICW3
	OUT DX,AL
	MOV AL,ICW4
	OUT DX,AL
	
	;从片初始化
	MOV DX,从片端口 0
	MOV AL,ICW1
	OUT DX,AL
	
	MOV DX,从片端口 1
	MOV AL,ICW2
	OUT DX,AL
	MOV AL,ICW3
	OUT DX,AL
	MOV AL,ICW4
	OUT DX,AL

8.2、可编程并行通信接口 8255A

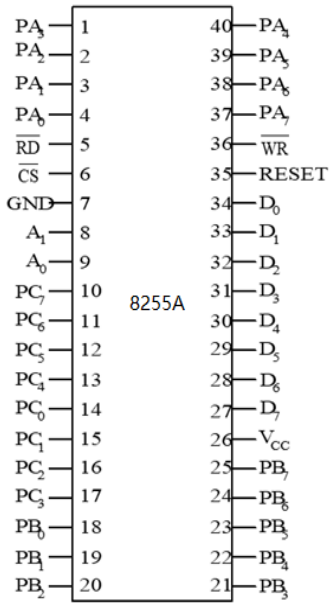
8.2.1、概述

8255A 是 Intel86 系列微处理机的配套并行接口芯片，它可为 86 系列 CPU 与外部设备之间提供并行输入/输出的通道。

8.2.2、内部结构



8255A 内部结构流程图



8255A 芯片引脚信号

端口 A 端口 B 端口 C	端口 A、端口 B 和端口 C 都是 8 位端口，可以选择作为输入或输出，还可以将端口 C 的高 4 位和低 4 位分开使用，分别作为输入或输出。 当端口 A 和端口 B 作为选通输入或输出的数据端口时，端口 C 的指定位与端口 A 和端口 B 配合使用，用做控制信号或状态信号。
A 组 控制部件 B 组 控制部件	这两组控制部件可以根据 CPU 送来的工作方式控制字控制 8255 工作方式的电路。 它们的控制寄存器接收 CPU 输出的方式控制字，由该控制字决定端口的工作方式，还可根据 CPU 的命令对端口 C 实现按位置位或复位操作。
数据总线 缓冲寄存器	这是一个双向 8 位三态数据缓冲器，8255A 正是通过它与系统数据总线相连，实现 8255A 与 CPU 之间的数据传送。 输入数据、输出数据、CPU 发给 8255A 的控制字等都是通过该部件传递的。
读 / 写 控制电路	负责管理 8255A 与 CPU 之间的数据传送过程。 它接收 $\overline{CS}$ 及地址总线的信号 A_1、A_0 和控制总线的控制信号 RESET、$\overline{WR}$、$\overline{RD}$，将它们组合后，得到对 A 组控制部件和 B 组控制部件的控制命令，并将命令送给这两个部件，再由它们控制完成对数据、状态信息和控制信息的传送。

8255A 读/写操作控制

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	操　作
0	1	0	0	0	写端口A
0	1	0	0	1	写端口B
0	1	0	1	0	写端口C
0	1	0	1	1	写控制寄存器
0	0	1	0	0	读端口A
0	0	1	0	1	读端口B
0	0	1	1	0	读端口C
0	0	1	1	1	无操作

8.2.3、工作方式

8.2.3.1、工作方式 0-基本输入/输出方式

方式 0 无须联络就可以直接进行 8255A 与外设之间的数据输入或输出操作。它适用于无须应答信号的简单的无条件输入/输出数据的场合，即输入/输出设备始终处于准备好状态。

在此方式下，A 口、B 口、C 口的高 4 位和低 4 位可以分别设置为输入或输出。

8.2.3.2、工作方式 1-选通输入/输出方式

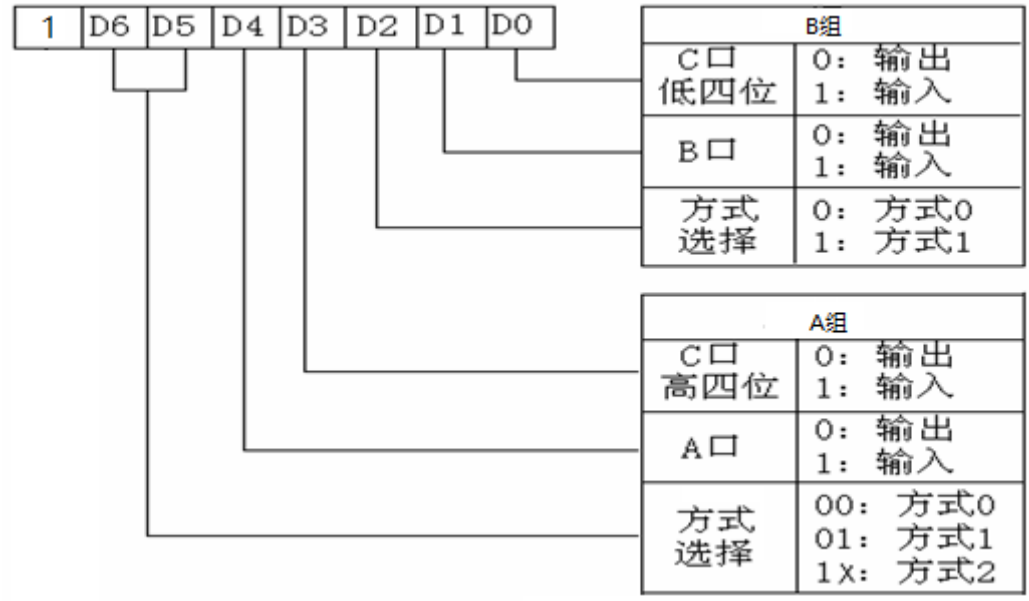
与方式 0 相比，它的主要特点是当 A 口、B 口工作于方式 1 时，C 口的某些 I/O 线被定义为 A 口和 B 口在方式 1 下工作时所需的联络信号线，这些线已经定义，不能由用户改变。

8.2.3.3、工作方式 2-选通双向传送方式

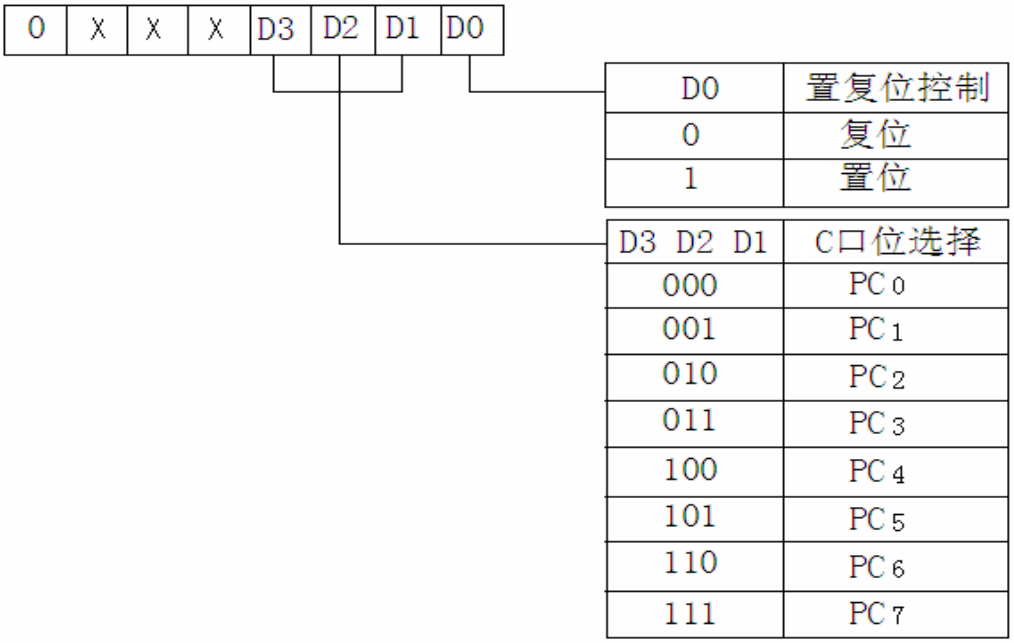
方式 2 即同一端口的 I/O 线既可以输入也可以输出，只有 A 口可工作于方式 2。

8.2.4、控制字

8.2.4.1、工作方式控制字

描述	8255A 的工作方式可由 CPU 写一个工作方式选择控制字到 8255A 的控制寄存器来选择
格式	

8.2.4.2、置位复位控制字

描述	8255A 的 C 口具有位控功能，即端口 C 的 8 位中的任一位都可通过 CPU 向 8255A 的控制寄存器写入一个按位置位/复位控制字来置“1”或清“0”，而 C 口中其它位的状态不变
格式	

8.2.5、初始化程序

方式选择	置位复位

MOV DX,0120H ;设置控制字寄存器地址
MOV AL,95H ;设置方式选择控制字
OUT DX,AL ;送入控制字寄存器中
	MOV DX,0120H ;设置控制字寄存器地址
	MOV AL,09H ;设置置位复位控制字
	OUT DX,AL ;送入控制字寄存器中

8.3、可编程计数/定时控制器 8253

8.3.1、概述

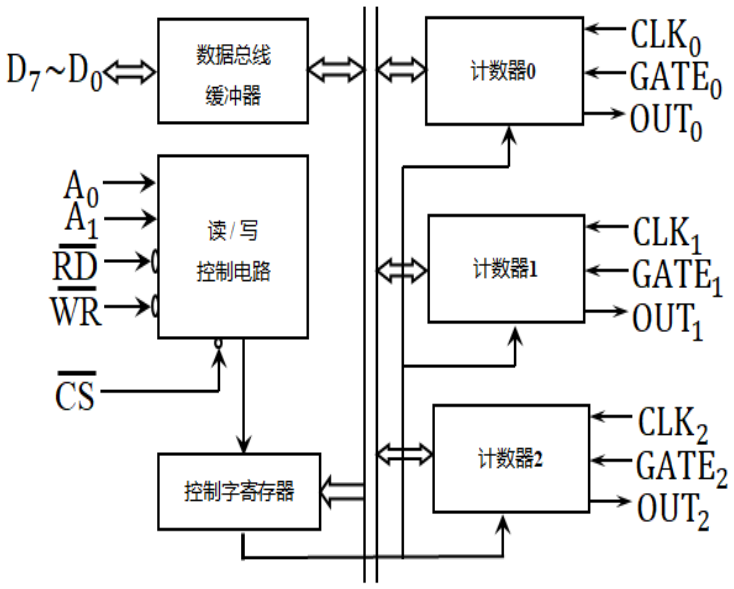
8253 可编程计数/定时控制器具有三个独立的通道，最高计数速率为 2.6MHz，分别称为计数器 0、计数器 1、计数器 2。

我们通常认为计数器和定时器完全一样。

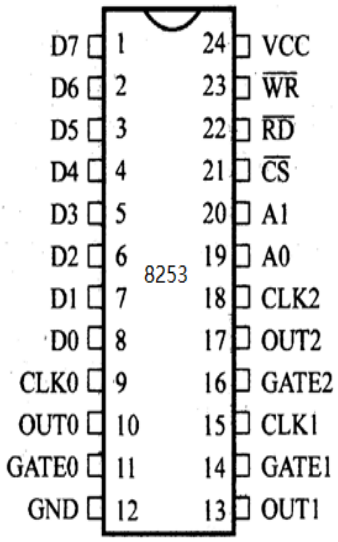
定时初值（定时系数）=输入频率/输出频率

定时初值（定时系数）=输入频率*输出周期

8.3.2、内部结构



8253 内部结构流程图



8253 芯片引脚信号

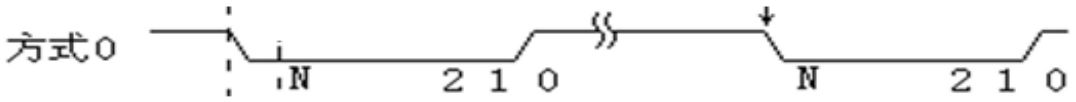
数据总线缓冲器	这是一个双向 8 位三态数据缓冲器，8253 正是通过它与系统数据总线相连，实现 8253 与 CPU 之间的数据传送。 输入数据、输出数据、CPU 发给 8253 的控制字等都是通过该部件传递的。
读 / 写控制电路	(1)、A_1A_0—端口选择信号，由 CPU 输入，进行端口选择。 (2)、$\overline{CS}$—片选信号，由 CPU 输入，低电平有效，通常由端口地址的高位地址译码形成。 (3)、$\overline{RD}$、$\overline{WR}$—读/写控制命令，由 CPU 输入，低电平有效。 $\overline{RD}$有效时，CPU 读取A_1A_0所选定通道内计数器的内容 $\overline{WR}$有效时，CPU 将计数值写入各个通道的计数器当中
控制器寄存器	控制字寄存器用来存放由 CPU 写入 8253 的方式选择控制字，由它来定义 8253 中各通道的工作方式。
计数器 0 计数器 1 计数器 2	8253 内部包含 3 个功能完全相同的通道，每个通道内部设有一个 16 位计数器，可进行二进制或 BCD 码计数。 若采用二进制计数时，则计数范围为：0000H~FFFFH，采用 BCD 码计数时，则计数范围为：0000~9999。与此计数器相对应，每个通道内设有一个 16 位计数值锁存器，必要时可用来锁存计数值。 要想计数次数最多，计数初值应该为 0；要想计数次数最少，计数初值应该为 1。 当某通道用作计数器时，应将要求计数的次数预置到该通道的计数器中，被计数的事件应以脉冲方式从CLK_i端输入，每输入一个计数脉冲，计数器内容减“1”，待计数值计到“0”，OUT_i端将有输出，表示计数次数到。 当某通道用作定时器时，由CLK_i输入一定频率的时钟脉冲。根据要求定时的时间长短确定所需的计数值，并预置到计数器中，每输入一个时钟脉冲，计数器内容减“1”，待计数值计到“0”，OUT_i将有输出，表示定时时间到。

8253 读/写操作控制

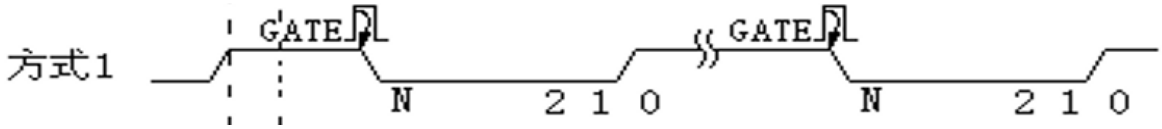
$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	操作
0	1	0	0	0	写计数器0
0	1	0	0	1	写计数器1
0	1	0	1	0	写计数器2
0	1	0	1	1	写方式控制字
0	0	1	0	0	读计数器0
0	0	1	0	1	读计数器1
0	0	1	1	0	读计数器2
0	0	1	1	1	无操作

8.3.3、工作方式

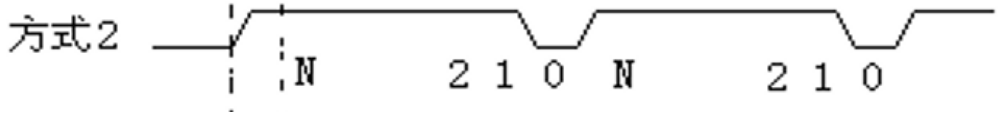
8.3.3.1、工作方式 0-计数结束中断



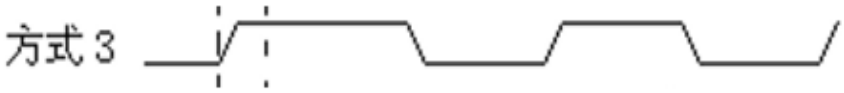
8.3.3.2、工作方式 1-硬件触发单拍脉冲



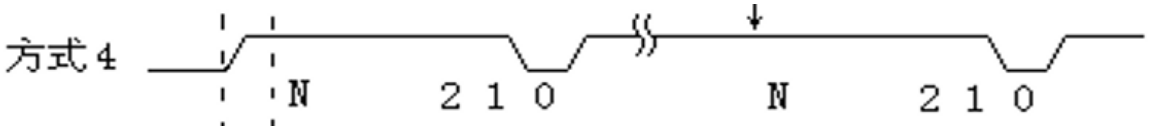
8.3.3.3、工作方式 2-频率发生器



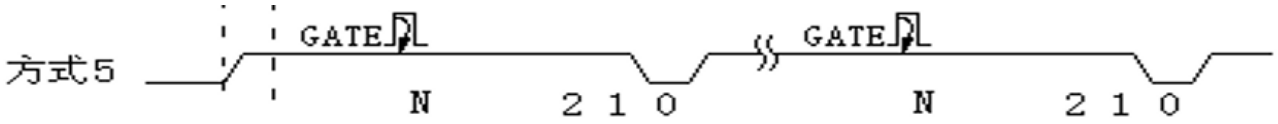
8.3.3.4、工作方式 3-方波发生器



8.3.3.5、工作方式 4-软件触发选通



8.3.3.6、工作方式 5-硬件触发选通



工作方式归纳如下：

(1)、硬件触发：方式 1、方式 5

(2)、软件触发：方式 0、方式 2、方式 3、方式 4

(3)、计数结束，输出负脉冲：方式 2、方式 4、方式 5

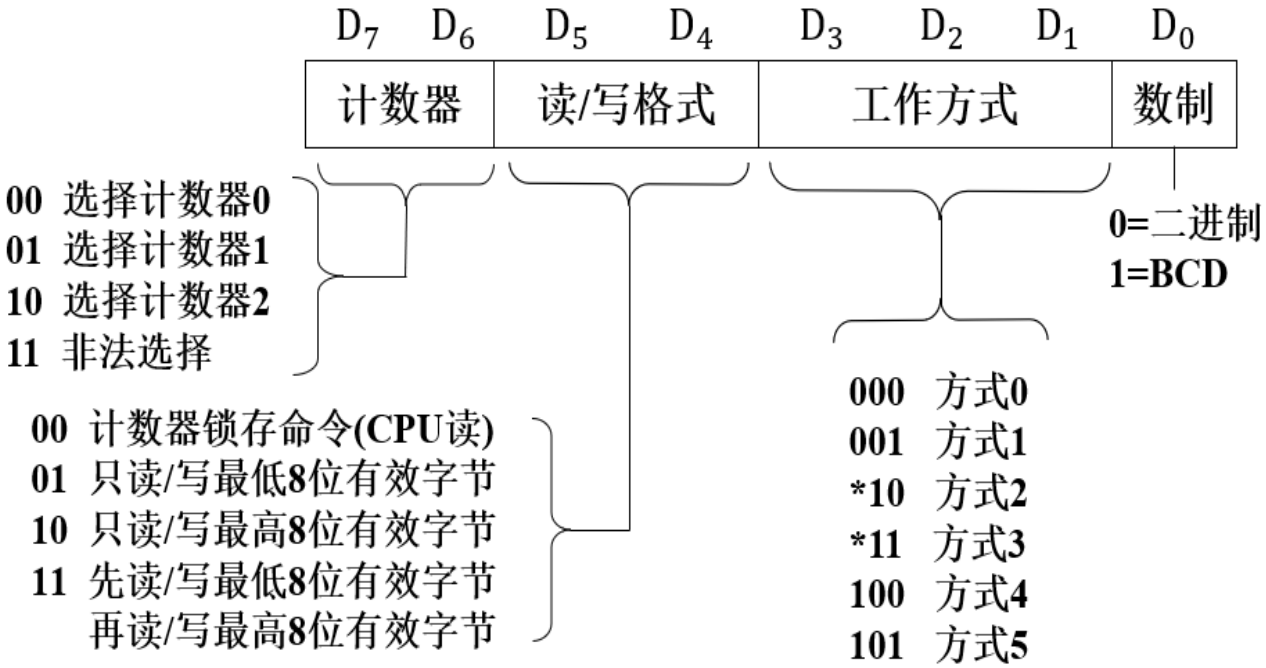
(4)、周期计数 / 既可以软件启动，又可以硬件启动：方式 2、方式 3

(5)、周期方波：方式 3

8.3.4、控制字

8.3.4.1、方式选择控制字

在 8253 的初始化编程中，由 CPU 向 8253 的控制字寄存器写入一个控制字来规定 8253 的工作方式。



8.3.5、初始化程序

计数初值为 1 个字节	计数初值为 2 个字节

MOV DX,控制字端口
MOV AL,10H ;设控制字
OUT DX,AL
 	
MOV DX,计数器端口
MOV AL,计数初值 ;设初始值
OUT DX,AL
	MOV DX,控制字端口
	MOV AL,0B4H ;设控制字
	OUT DX,AL
	
	MOV DX,计数器端口
	MOV AX,计数初值
	OUT DX,AL ;设低 8 位
	MOV AL,AH
	OUT DX,AL ;设高 8 位

常见的换算单位:

1s=1×10^3ms（毫秒）	1GHz=1×10^9Hz
1s=1×10^6us （微妙）	1MHz=1×10^6Hz
1s=1×10^9ns （纳秒）	1KHz=1×10^3Hz