微机系统与接口

Leo

2024 年 3 月 19 日

目录

第一章　8086CPU

1.1　计算机系统组成

1.1.1　计算机组成

冯诺依曼结构的计算机分为外设和主机。外设就是输入输出设备，主机就是 CPU（包括运算器和控制器）以及存储器

工作原理

1. 从输出设备将数据或指令送到运算器
2. 运算结果送给存储器或输出设备
3. 从存储器取出数据到运算器再运算
4. 从存储器取出指令送给控制器，译码分析后产生各种命令（上述动作都是由指令产生的各种命令控制的）

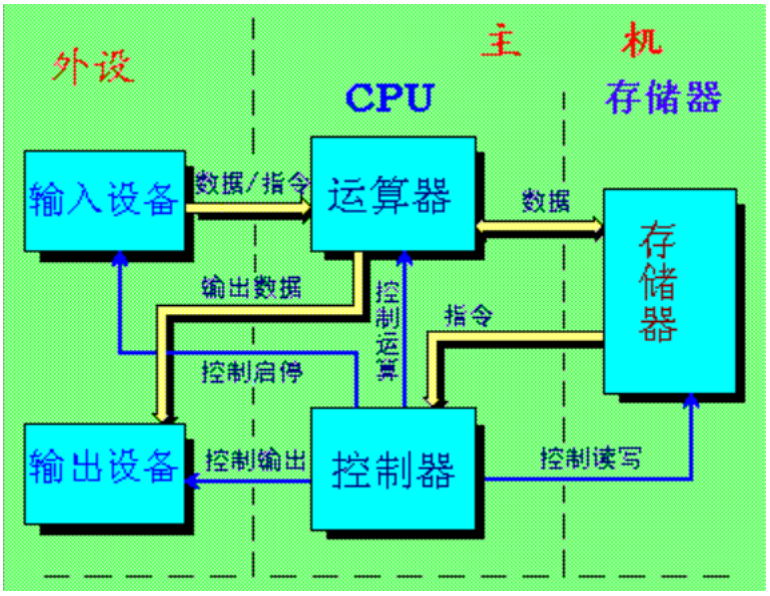

图 1.1: 冯诺依曼计算机体系结构图

1.1.2 微机组成

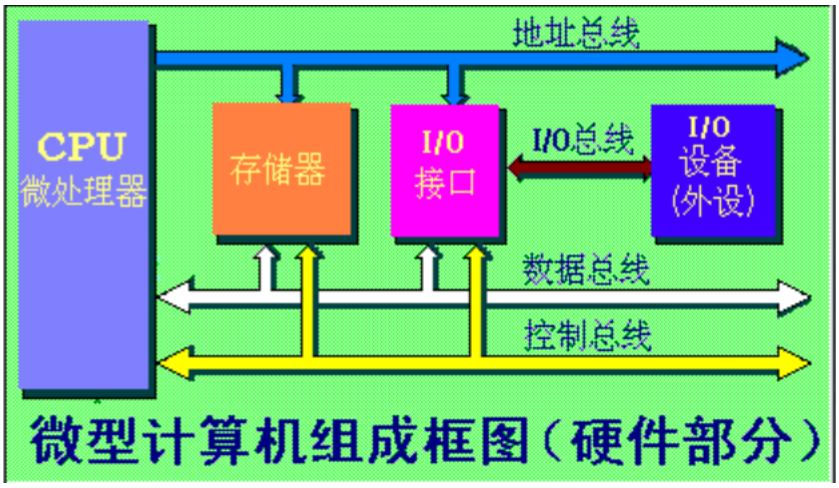

图 1.2: 微机组成框图

微型计算机由 CPU 微处理器、存储器、IO 接口和 IO 设备（外设）组成有 4 种总线

1. 地址总线：连接 CPU 和存储器与 IO 接口
2. 数据总线：连接 CPU 和存储器与 IO 接口
3. 控制总线：连接 CPU 和存储器与 IO 接口
4. IO 总线：连接 IO 接口和 IO 设备

工作过程

1. 输入设备把指令送入 IO 接口，经过数据总线送入指定存储单元，存储器的地址由地址总线给出
2. CPU 从存储器取指令，产生地址、数据、控制信号，分别可以选中存储单元或端口、与选中的单元或接口交换数据、控制其他信号

下面细讲各组成部分

CPU

即中央处理单元，计算机所有操作均在 CPU 的控制下进行，CPU 型号直接决定了计算机的档次

Intel：

1. 4004:4 位微机
2. 8085/8080：8 位
3. 8086/8088：16/准 16 位 PC/XT 机
4. 80286：16 位 PC/XT 机

5. 80386/80486：32 位机

6. Pentium：奔腾 32/64 位机

Motorola：6800（8 位）——68000（16 位）
Zilog：Z80（8 位）——Z8000（16 位）

总线：BUS

按传送的信息类型可以分为 3 类

1. 数据总线：双向传送数据信息。可以连接多个设备，但是同一时间只能有一个设备在总线上（通过三态门控制）N 根数据线可以同时传送 N 位信息。8086 有 16 根，8088 内部 16，外部 8 根

2. 地址总线：单向，用来寻址内存单元或 IO 端口。8086/8088 都是 20 根

3. 控制总线：输入输出，传送控制信号

按规模、用途和应用场合可以分为 3 类

1. 片级总线或元件级总线：用于芯片一级的互联，如 CPU 和存储器，CPU 和 IO 接口

2. 外部总线：也称为通信总线，用于微机和其他电子设备之间的连线

3. 系统总线：也成为内总线和板级总线，用于微机中各插线板之间的连线

存储器：Memory

用来存放数据和程序，每个存储单元存放一个字节，即 8 位 bit。对每个存储单元编一个号，称为地址，地址单元和内容常常用 16 进制表示存储器中包含的字节数称为存储容量

图 1.3: 存储器结构举例

8086 有 20 根地址线，可以直接寻址 $2^{20} = 1M$ 内存单元。使用了段加偏移的方法，一个物理地址有 20 位，可以表示为“段地址：偏移地址”= 段地址 *16+ 偏移地址

8086 存储器

一个程序可以有 4 个段，段名和段寄存器对应为

1. 代码段 CS：code segment 将要执行的下一条指令一定是从 CS:IP 指示的单元中取出
2. 数据段 DS：data segment
3. 堆栈段 SS：stack segment
4. 附加段 ES：extra segment

IO 接口

为什么要有接口？因为外设与 CPU 交换数据的时候会带来一些问题

1. 速度不匹配，一般是 CPU 远快于外设
2. 信号电平不匹配
3. 信号格式不匹配
4. 时序不匹配
5. 译码问题

接口电路主要实现的功能

1. 数据缓冲，使速度匹配：8255A 通用并行接口、锁存器
2. 电平转换：MC1488，MC1489，MAX233 电平转换电路
3. 串并行转换：8251A，8250 串行接口芯片
4. 数模模数转换：DAC0832，ADC0809
5. 地址译码：74LS138
6. 中断控制：8259A 中断控制器
7. 存储器直接数据传送控制：8237ADMA 控制器
8. 定时计数功能：8253，8254 计数器/定时器

外设（即 IO 设备）

输入设备：键盘、鼠标器、扫描仪、CD-ROM 驱动器；
输出设备：CRT 显示器、LPT 打印机、扬声器、LED/LCD 显示器

1.1.3 微机系统组成

- 硬件
  - 主机
    - CPU
    - 存储器
    - 各类接口芯片
  - 外设
    - 输入设备
    - 输出设备
- 软件
  - 系统软件
    - 操作系统：DOS、WINDOWS、UNIX
    - 汇编程序
    - 解释程序（对 BASIC）
    - 编译程序（对高级语言）
  - 程序设计语言
    - 机器语言(二进制码，计算机只认得这种语言)
    - 汇编语言（符号语言）
    - 高级语言（C、FORTRAN、PASCAL）
  - 应用软件
    - 软件包
    - 数据库
    - 文字处理软件、娱乐、教育等

(1.1)

1.2 8086CPU 系统

1.2.1 内部结构

8086 是 16 位，8088 是准 16 位微处理器，都具有 16 位地址总线，可处理 8 或 16 位数据；外部都具有 20 根地址总线，可以直接寻址 $2^{20} = 1M$ 字节的内存空间，可以寻址 $2^{16} = 64K$ 个 IO 端口。

采用 +5V 供电，单相时钟

8086 外部有 16 根地址总线，8088 有 8 根，与外部传送数据时，都要执行外部总线周期。

8086 结构与原理

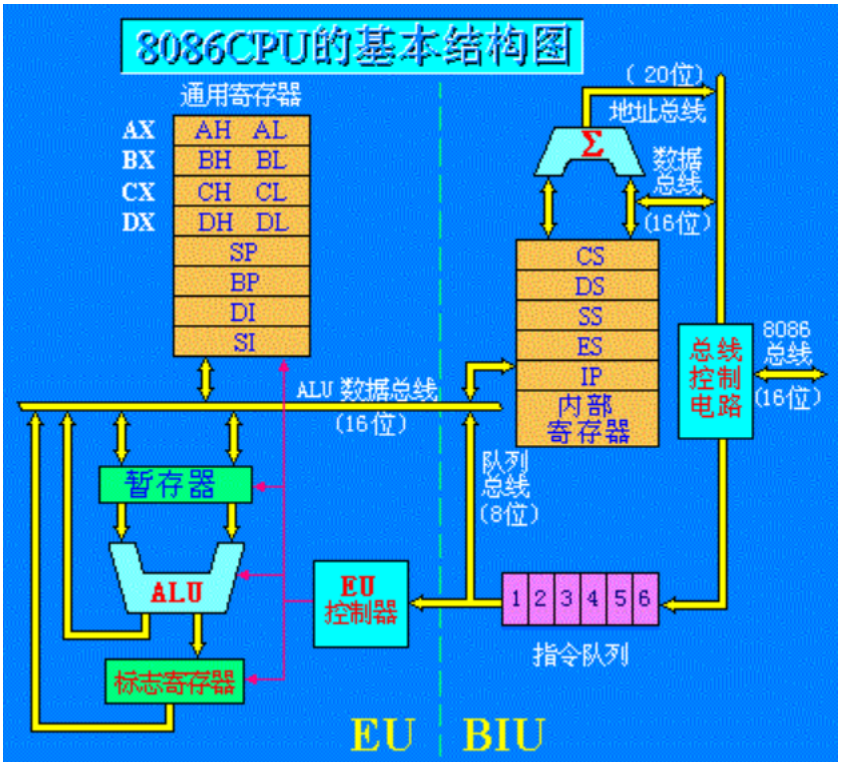

图 1.4: 8086CPU 基本结构

主要由两部分组成：
EU 执行单元

1. 负责执行全部指令
2. 给 BIU 提供数据和地址信息
3. 管理通用寄存器和标志寄存器，不直接和外部打交道

BIU 总线接口单元

1. 执行所有的外部总线周期
2. 负责与存储器，IO 端口打交道

工作流程

1. 读操作：BIU 根据 CS：IP 在地址加法器形成 20 位物理地址，通过总线控制电路从 mem 中取出指令，送到 6 字节的指令队列中等待执行
2. EU 从队列中取指令，译码后执行。此时没有用到总线，BIU 可继续工作，把队列填满。如果 EU 没有向 BIU 申请读或写存储器或 IO 操作时，BIU 空闲（* 在执行指令时预先取下一条指令的技术称为流水线）
3. 若遇到 JMP 或 CALL 指令，队列中的内容作废，从新的转移地址中取指令码

4. ALU 完成算数/逻辑运算后，将运算结果通过数据总线送到 EU 的通用寄存器或暂存器；或者送到 BIU 的内部寄存器再送到存储器或 IO 端口。本次操作的状态反映在标志寄存器 FLAGS 中

8088 内部结构

与 8086 基本一样，主要区别在于：

1. 8088 指令队列为 4 字节，而不是 6 字节
2. 8088 外部数据总线为 8 位而不是 16 位

总线周期简述

总线周期：在 8086 中，BIU 完成一次访问存储器或 IO 端口的时间称为总线周期。最基本的总线周期由 T_1 到 $T_4$4 个时钟周期组成，也称为 T_1 到 T_4 状态。

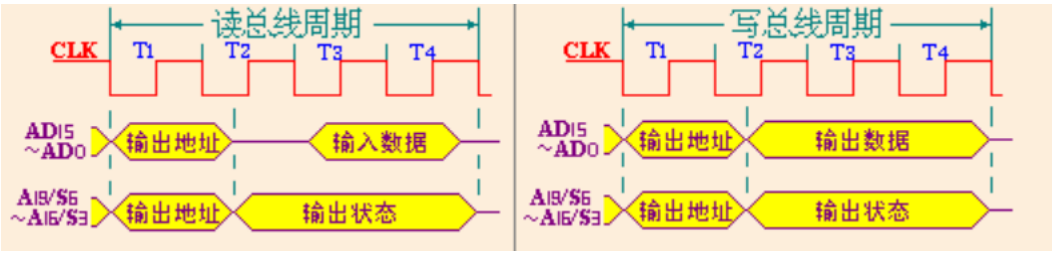

图 1.5: 读周期和写周期

读周期：

1. T_1 状态通过 4 条地址/状态复用线和 16 条地址数据复用线（共 20 条线）向存储器输出指定地址
2. T_2 进入高阻态，缓冲一下，让 AD 地址数据复用线从输出地址变为输入数据信号。但是地址/状态复用线从 T_2 到 T_4 一直输出状态信号
3. T_3T_4 16 条地址数据复用线接收指定单元内容

内部寄存器简述

- 通用寄存器
  - 数据寄存器
    - 累加器 AX
    - 基址寄存器 BX
    - 计数寄存器 CX
    - 数据寄存器 DX
  - 指针和变址寄存器：涉及到存储器操作数时，可用的寄存器
    - 堆栈指针 SP：不能做通用寄
    - 基址指针 BP
    - 源变址指针 SI
    - 目的变址指针 DI

(1.2)

通用寄存器：16 位寄存器用来存放数据和地址，8 位的只能存放数据

数据寄存器隐含用途：

1. AX: 在字 IO 指令中，做 16 位数据寄存器；在字节乘法中存放乘积；在字节除法中存放被除数
2. AH：字节除法中存放余数
3. AL：字节 IO 指令中做 8 位数据寄存器；在字节乘法中存放乘数；在字节除法中存放商；BCD 运算中必须将运算结果或被除数放在 AL 中才能用二-十进制调整指令调整；在 XLAT 指令中做指针和存放代码转换结果
4. BX：做基地址指针；XLAT 中存放表头地址
5. CX：字符串操作和循环操作中做计数器
6. CL：在算数逻辑移位和循环移位指令中做移位次数计数器
7. DX：字乘法中存乘积的高半部分；字除法中存放被除数的高半部分和余数；在间接寻址的 IO 指令中做端口地址寄存器

指针和变址寄存器隐含用途：

1. SI：在字符串指令中做源串偏移地址指针
2. DI：在字符串指令中做目的串偏移地址指针
3. SP：在堆栈操作中指示栈顶位置
4. BP：在堆栈操作中指示堆栈中一个数据区的偏移地址，可用于访问堆栈中的某个数据

段寄存器：

计算机存储器中存放三类信息：代码信息，数据信息和堆栈信息，这三类信息存放在各自的内存区域中，用段寄存器来指示这些段的起始地址的高 16 位部分，一共有 4 种段寄存器1.1.2

段寄存器和 BX、BP、SI、DI 等配合，可以构成多种寻址方式。例如 CS：IP 指出下一条要执行的指令的地址。SS：SP 指向堆栈栈顶单元

标志寄存器：

16 位 FLAGS 设置了 9 个标志位，其中 6 个为状态标志，用来反映上次运算结果的状态。另外三个为控制标志，用来控制 CPU 的某些操作

1. OF(overflow) 溢出标志：用于带符号数的运算，如果两个正数相加出现负数或者负数相加出现正数，则为 1.（正确的结果也可能有进位，但是进位位自然丢失或给更高位）
2. DF(direction) 方向标志：控制 SI、DI 的变化方向。DF=0，则 SI、DI 自增；反之自减
3. IF(interrupt) 中断标志：STI 指令使 IF=1，允许 CPU 响应可屏蔽中断。CLI 指令使 IF=0，禁止 CPU 响应可屏蔽中断
4. TF(trap) 陷阱标志：TF=1，CPU 处于单步工作方式，每执行一条指令自动内部中断，并打印信息，便于调试
5. SF(sign) 符号标志：数字的最高位 MSB=1 时，SF=1，表示结果为负值；MSB=0 时，SF=0，表示结果为正值
6. ZF(zero) 零标志：运算结果为 0 时 ZF=1
7. AF(auxiliary) 辅助进位标志:8 位运算时,低半字节向高半字节有进位或借位时 AF=1.16 位运算时仅在低字节部分使用。一般用于 BCD 码运算的十进制调整（加 6 或减 6）
8. PF(parity) 奇偶标志：算数逻辑运算时，结果有偶数个 1 时则 PF=1，一般用来检验数据传输中的错误
9. CF(carry) 进位标志：最高位 D_7 或 D_{15} 产生进位或借位时 CF=1

指令指针

IP 指针用来存放将要执行的下一条指令相对于现行代码段的偏移地址，用来控制指令序列的执行流程。

程序运行时，BIU 自动对 IP 进行修改，使它始终指向下一条指令的地址。程序不能直接访问 IP，只能由 CPU 自动控制

1.2.2 存储器基础

存储器主要用来存放程序和数据，存储容量和存取速度是决定性指标。内存容量一般指 RAM 大小

存储器分类

$$\begin{cases} \text{内存} \begin{cases} RAM \begin{cases} SRAM \\ DRAM \end{cases} \\ ROM \begin{cases} ROM \\ PROM \\ EPROM \\ E2PROM \end{cases} \end{cases} \\ \text{外存} \begin{cases} \text{硬盘} \\ \text{软盘} \\ \text{磁带} \\ \text{光盘} \end{cases} \\ \text{高速缓冲存储器 Cache} \end{cases} \tag{1.3}$$

8086 分段概念

为什么要分段？因为地址线有 20 根，但是 CPU 寄存器只有 16 位。

怎么分段？一个现行程序可以分为 DS、CS、SS 和 ES 四段，凡是能被 16（或 10H）整除的地址处均可分段。如 00000H，00010H，00150H 等

各段之间可以相互独立也可以相互覆盖

分段的好处？

1. 大部分指令不涉及段寄存器的值，仅改变偏移量，这样可以缩短指令长度，提高运算速度
2. 内存分段为程序的浮动装配创造条件

8086 存储器结构

8086 可直接寻址的 1MB 内存分成 2 个 512K 的存储体，$A_0 = 0$ 选到偶地址体，$A_0 = 1$ 选到奇地址体。

CPU 访问存储器都是从偶地址体开始的，以字为单位进行操作。读 00000 字时，从 00000 开始读两个字节，读出来的是 (00001)(00000) 代表的字；读 00003 字时，从 00004 开始读到 00003，即读出内容为 (00004)(00003) 代表的字。

8086 系统与存储器的连接

1. 每个存储器单元有 8 根数据线，奇地址体与 CPU 的高 8 位相连，偶地址体与低 8 位相连
2. 用 A_0 和 $\overline{BHE}$(Bus High Enable) 来控制地址体的选中与否

$\overline{BHE}$	A_0	操作
0	0	奇偶地址体都被选中，读字
0	1	读偶地址体字节
1	0	读奇地址体字节
1	1	无操作

8086 堆栈操作

一个堆栈不超过一个段的指示范围（$2^{16} = 64K$）, 由 SS 给出段基地址，SP 存放偏移地址，指示栈顶位置，$SP \leq FFFEH$

堆栈的基本处理单元是字，压栈时，一个字的高 8 位先入栈，低 8 位后入栈，SP 的值减二（注意 16 进制减法的不同）

1.2.3 引脚功能

8086CPU 引脚

$AD_{15} - AD_0$ **地址/数据总线** 分时复用：在总线周期的 T_1 状态，传送存储器或 IO 端口的地址信息，然后在 ALE 信号控制下用锁存器将 $A_{15} - A_0$ 锁存住

T_2 状态：读总线周期下为高阻态，写总线周期下，传送数据信号 $D_{15} - D_0$

$T_3 - T_4$ 状态：传送传送数据信号 $D_{15} - D_0$

INTR 中断请求 为高电平时，表示外部向 CPU 发送中断请求信号。CPU 在每条指令的最后一个时钟周期都要对其信号进行采样，若为 1，CPU 的 IF=1，则 CPU 转入中断响应周期

NMI 不可屏蔽中断 CPU 一旦检测到 NMI 的上跳变信号，则执行完当前指令后，进入类型 2 的不可屏蔽中断。这类中断不受 IF 影响，不能用软件屏蔽。

CLK 由外部时钟信号 8284 提供，基本在 MHz 级别

$A_{19}S_6 - A_{16}S_3$ **地址/状态总线** 分时复用：在总线周期的 T_1 状态做地址线用，可用于访问存储器，但是访问 IO 端口只用到低 16 位地址线，所以这四根线无效

$T_2 - T_4$ 状态：输出状态信息

1. $S_6 = 0$
2. $S_5 = 1$ 允许可屏蔽的中断请求，$S_5 = 0$ 禁止可屏蔽的中断请求
3. S_4S_3 段寄存器状态线，从 00 到 11 分别表示当前正在使用的段寄存器为 ES,SS,CS,DS

$\overline{BHE}/S_7$ **总线高位有效/状态线** 分时复用：T_1 输出有效的 $\overline{BHE}$ 信号。其他时候输出 S_7，在 8086 中 S_7 无意义

$MN/\overline{MX}$ **最小/最大模式**

$\overline{RD}$ **读信号** 执行读指令时该引脚变为低电平，到底是读存储器还是读 IO 取决于 $M/\overline{IO}$ 信号

READY 准备好信号 由 8284 时钟信号产生器产生，一般 CPU 与低速外设交换数据时要等外设，此时 READY 变低电平，这样可以在 T_3 后插入若干等待周期 T_w；READY 变为高电平后，下一个周期进入 T_4

$\overline{TEST}$ **测试信号** CPU 执行 WAIT 指令时，每隔 5 个 T 就检测一次该引脚，若为高电平（无效）则进入空闲周期，反之继续执行下一条指令
RESET 复位信号 高电平有效，至少维持 4 个时钟周期。复位时：

1. CPU 立即中止所有操作，总线无效
2. IP,DS,ES,SS,FLAGS 清零，CS=FFFFH，复位结束后，CS:IP=FFFF:0000H,CPU 从这里开始执行程序，所以可在这里安排一条 JMP RESET
3. 指令队列清空

8088CPU 引脚

$A_{15}-A_8$ **地址线** 不与数据线复用，因为 8088 只需要 8 根数据线
SS_0**(High)** 最小模式下为状态信号，与 $IO/\overline{M}$,$DT/\overline{R}$ 一起决定现行总线周期的状态。最大模式下接高电平。
$IO/\overline{M}(S_2)$ 在 8088 中为 1 时访问 IO，为 0 时访问存储器。在 8086 中反之

1.2.4 工作模式

$MN/\overline{MX}$ 接高电平时工作于最小模式：

1. 系统中只有一个 8086 或 8088 微处理器
2. 所有总线控制信号均由 CPU 直接产生或接收

$MN/\overline{MX}$ 接低电平时工作于最大模式：

1. 系统中允许多个处理器共同工作
2. 有的控制信号由 CPU 直接产生，有的由 8288 总线控制器译码后产生
3. 单个 CPU 也可以工作在最大模式

两者的共同点：

1. 都需要用地址锁存器 8282/8283 或 74LS373(因为 20 位地址线，所以都要用三片芯片)，使地址数据线先传送地址并锁存，再传送数据信号
2. 都需要使用 8286/8287 或 74LS245（8086 用两片，8088 用一片）控制数据传送的方向，同时增强数据总线的驱动力，在小系统中可以不用数据收发器。

8086 最大模式

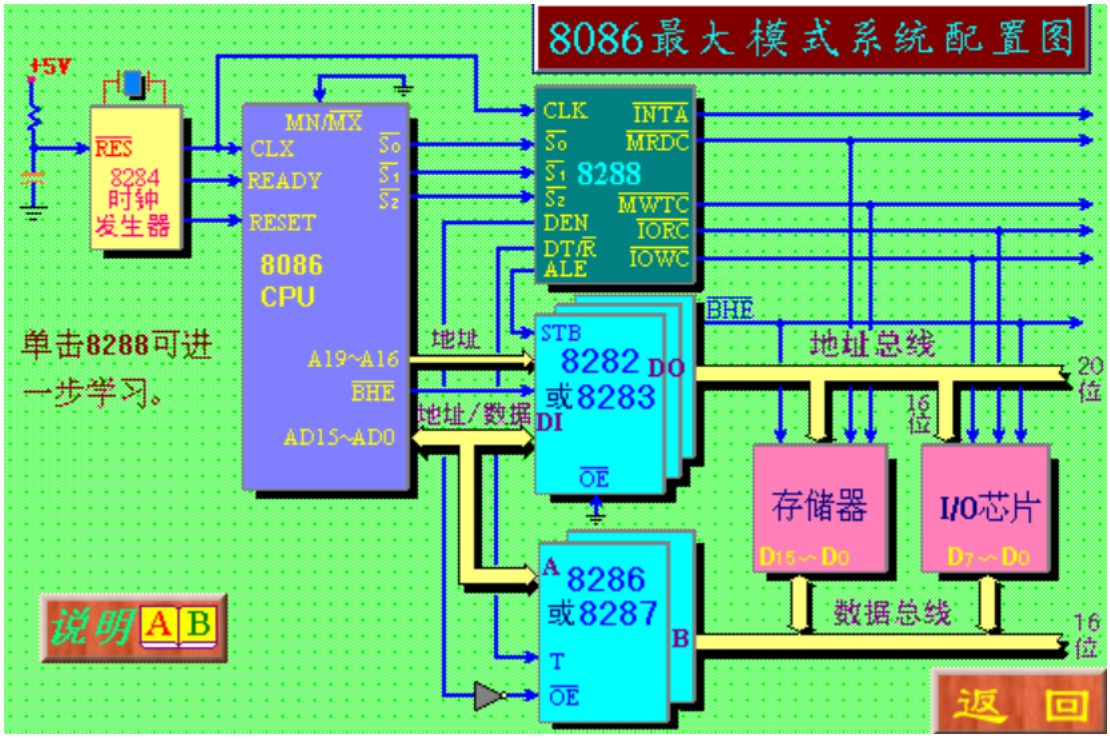

图 1.6: 8086 最大模式系统配置图

特征：

1. $MN/\overline{MX}$ 接地
2. 增加总线控制器 8288 来产生原来由 CPU 产生的控制信号
3. 8288 产生的 ALE（地址锁存允许）、DEN（数据传送允许）和 $DT/\overline{R}$(数据收发允许) 三种信号与 CPU 一致

8288 总线控制器 接收时钟产生器送出的 CLK 信号，其频率与 CPU 的 CLK 相同。同时还接收 CPU 送来的状态信号，从输出端输出各种控制信号

1. $DT/\overline{R}$,ALE: 与最小模式信号相同
2. DEN：与最小模式的 $\overline{DEN}$ 相位相反
3. $\overline{MRDC}$: 存储器读，相当于 $\overline{RD}, M/\overline{IO}$ 结合，选中存储器。引到系统总线上命名为 $\overline{MEMR}$
4. $\overline{MWTC}$: 存储器写，相当于 $\overline{WR}, M/\overline{IO}$ 结合，选中存储器。引到系统总线上命名为 $\overline{MEMW}$
5. $\overline{IORC}$:IO 读，相当于 $\overline{RD}, M/\overline{IO}$ 结合，选中 IO 端口。引到系统总线上命名为 $\overline{IOR}$
6. $\overline{IOWC}$:IO 写，相当于 $\overline{WR}, M/\overline{IO}$ 结合，选中 IO 端口。引到系统总线上命名为 $\overline{IOW}$
7. $\overline{INTA}$: 中断响应信号

8. *$\overline{AIOWC}$: 超前写 IO 命令，与 $\overline{IOWC}$ 相比，8288 提前一个时钟周期向 IO 端口发出写命令。可以使较慢的设备提前一个周期进入写操作。用于 PC 机中。

9. *$\overline{AMWC}$: 超前写存储器命令，与 $\overline{MWTC}$ 相比，8288 提前一个时钟周期向存储器发出写命令

8086 最小模式

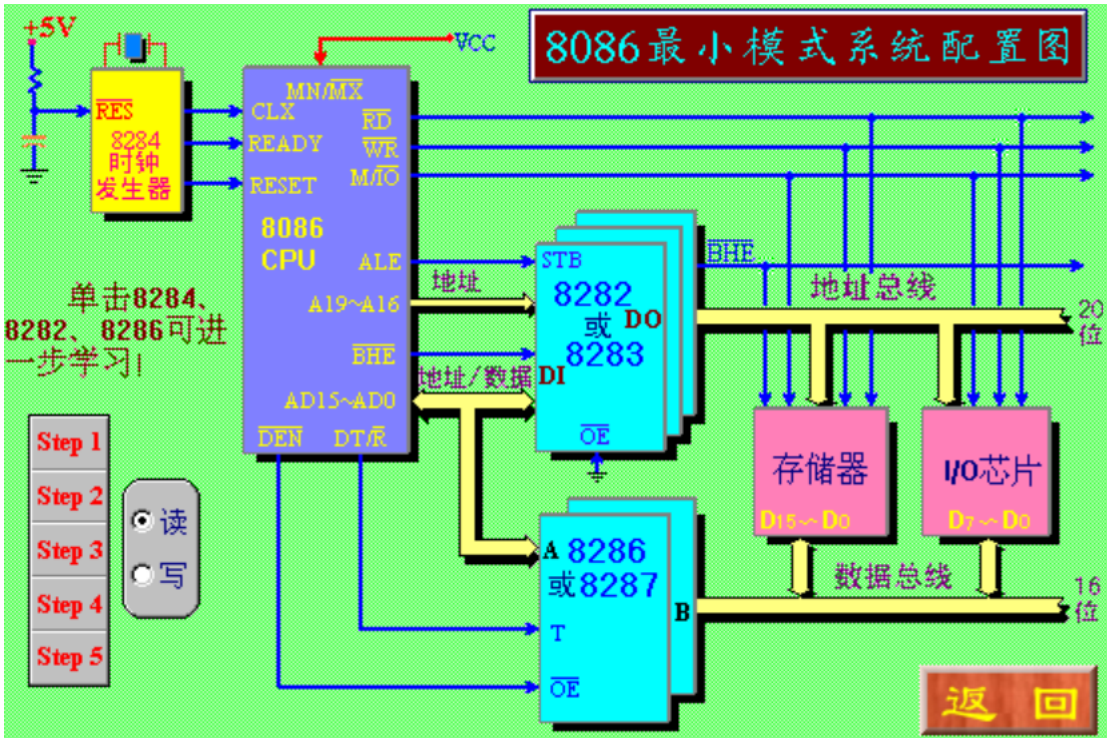

图 1.7: 8086 最小模式系统配置图

读存储器操作流程 (读 IO 仅在第一步 $M/\overline{IO}=0$ 处不一样)

1. $M/\overline{IO}$ 置 1 选中存储器，$DT/\overline{R}$ 置零，使 8286T 引脚为零，使 CPU 准备接收数据

2. 地址信号和 $\overline{BHE}$ 输出给 8282，同时送出一个 ALE 正脉冲，在 ALE 下降沿锁存

3. 三态门打开，8282 输出地址信号和 $\overline{BHE}$ 到存储器某一指定单元

4. CPU 发出 $\overline{RD}=0$ 读信号，同时 CPU 的 $\overline{DEN}=0$ 使 8286 可以接收数据，且 8286 的 T=0，允许接收数据

5. 从选中的存储器单元中读出来送给 8286，再送给 CPU 的 $AD_{15}-AD_0$

写存储器操作流程 (写 IO 仅在第一步 $M/\overline{IO}=0$ 处不一样)

1. $M/\overline{IO}$ 置 1 选中存储器，$DT/\overline{R}$ 置 1，使 8286T 引脚为 1，使 CPU 准备发送数据

2. 地址信号和 $\overline{BHE}$ 输出给 8282，同时送出一个 ALE 正脉冲，在 ALE 下降沿将 20 位地址和 $\overline{BHE}$ 锁存

3. 8282 三态门打开，8282 输出地址信号和 $\overline{BHE}$ 到存储器某一指定单元

4. CPU 发出 $\overline{WR}=0$ 读信号，同时 CPU 的 $\overline{DEN}=0$ 使 8286 可以接收数据，且 8286 的 T=1，允许发送数据

5. CPU 通过 $AD_{15}-AD_0$ 把要写入的数据送给 8286，再送到存储单元

8282 地址锁存器（或用 74LS373）

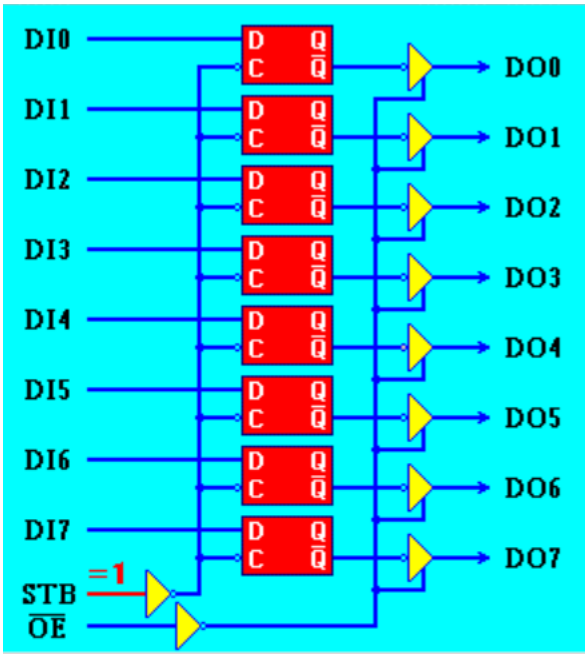

图 1.8: 8282 地址锁存器

由 8 个 D 触发器和 8 个三态门组成，有 STB 和 $\overline{OE}$ 两个控制端。STB=1 时 D 触发器透明，STB 产生下降沿时，锁存，STB=0 时 D 端变化 Q 不变；当 $\overline{OE}=0$ 时三态门开启，可以把 $\overline{Q}$ 反相后输出，相当于输出 Q，当 $\overline{OE}=1$ 时不能输出

8286 数据收发器（或用 74LS245）

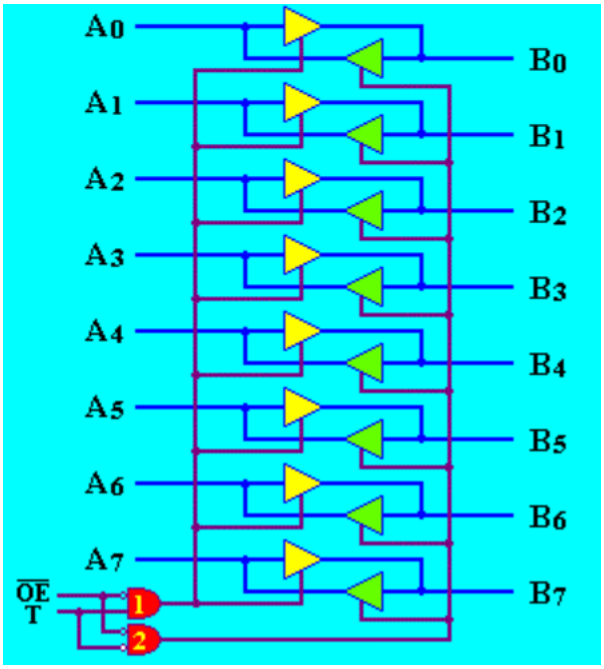

图 1.9: 8286 数据收发器

由 8 组首尾相连的反相器组成。$\overline{OE}=1$ 时禁止收发数据，输出低电平；$\overline{OE}=0$ 时允许收发数据，若 T=1，传送方向为 A 到 B（发送），反之为接收。

8284 时钟产生器

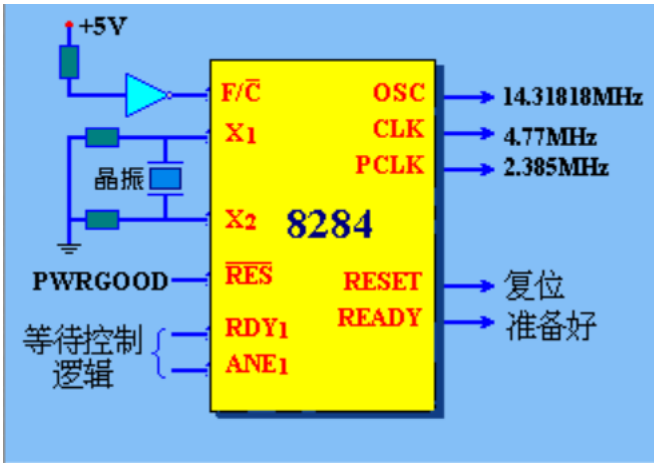

图 1.10: 8284 时钟发生器

1. $F/\overline{C}=0$ 时，输入频率由晶振决定，可以输出三种脉冲信号
2. $\overline{RES}$ 接 PWRGOOD，系统加电，四种直流输出电压（$\pm5,\pm12$）正常后，送出 PWRGOOD，经 8284 同步，产生 RESET 送到 CPU 的 RESET 引脚引起复位
3. RDY、AEN 与外部等待逻辑电路相连，经 8284 同步后可以产生 READY 信号送给 CPU，以产生等待周期 T_w

8088 最大模式

地址是 20 位，但无 $\overline{BHE}$ 信号，数据线只有 8 位，所以 8286 只用一片。地址/数据复用线为 $AD_7 - AD_0$

8088 最小模式

也要 3 片 8282，在小系统中可以不用 8286

1.2.5 典型时序

读总线周期

总线周期分为 4 个时钟周期，都是在时钟的下降沿触发。

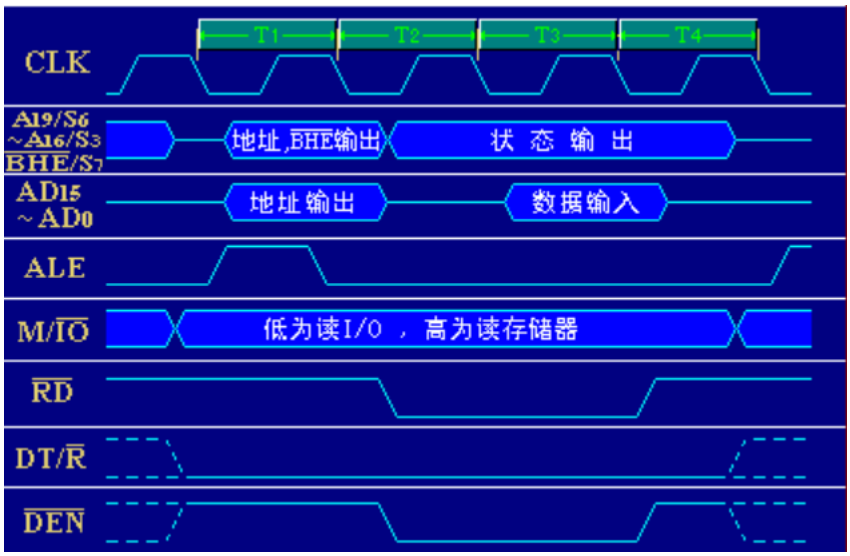

图 1.11: 读总线周期

写总线周期

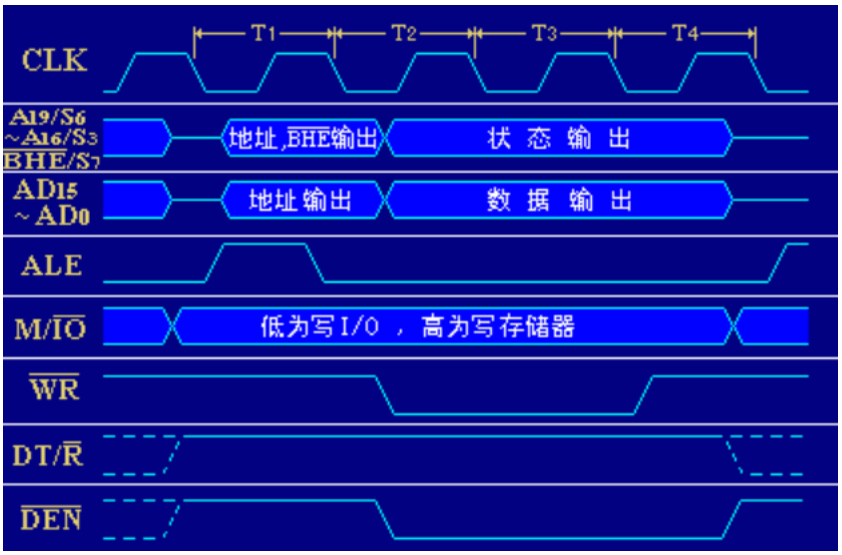

图 1.12: 写总线周期

1.2.6 存储器设计

1.2.7 SRAM 6264

容量大小：8K×8
引脚及功能：

1. $A_{12} - A_0$:13 根地址线，选择芯片中 2^{13} 个存储单元中的任一个单元
2. $I/O_7 - I/O_0$: 八根双向数据线，并行送数据
3. $\overline{WE}$: 写入允许信号
4. $\overline{OE}$: 读出允许信号
5. $\overline{CE_1}, \overline{CE_2}$: 片选信号，两者均为低电平才能对芯片进行读写操作。其中一个作奇偶选择信号用。

1.2.8 SRAM 2114

容量大小：1K×4
引脚及功能：

1. $A_9 - A_0$:13 根地址线，选择芯片中 2^{13} 个存储单元中的任一个单元
2. $I/O_4 - I/O_0$: 八根双向数据线，并行送数据
3. $\overline{WE}$: 写入允许信号
4. $\overline{CS}$: 片选信号

1.2.9 EPROM 2764

EPROM 允许用户将写入的内容用专门的擦除器擦除，允许反复擦除与重写。但是 2764 在正常使用时只能读出。容量大小：8K×8

引脚及功能：

1. $A_{12} - A_0$:13 根地址线，选择芯片中 2^{13} 个存储单元中的任一个单元
2. $O_7 - O_0$: 八根双向数据线，并行送数据
3. $\overline{OE}$: 读出允许信号
4. $\overline{CE}$: 片选信号
5. PGM: 编程脉冲控制
6. V_{pp}: 编程电压输入
7. V_{cc}: 工作电压，接 +5V

* 编程方式时的引脚状态

1. $A_{12} - A_0$: 选中存储单元，逐字编程
2. $\overline{OE}$: 接 +5V
3. $\overline{CE}$: 接 +5V
4. PGM: 对每个单元编程时，从该引脚上输出一个 50ms 宽的正脉冲
5. V_{pp}: 接 +21V 到 +25V
6. V_{cc}: 工作电压，接 +5V

1.2.10 存储器系统连线图

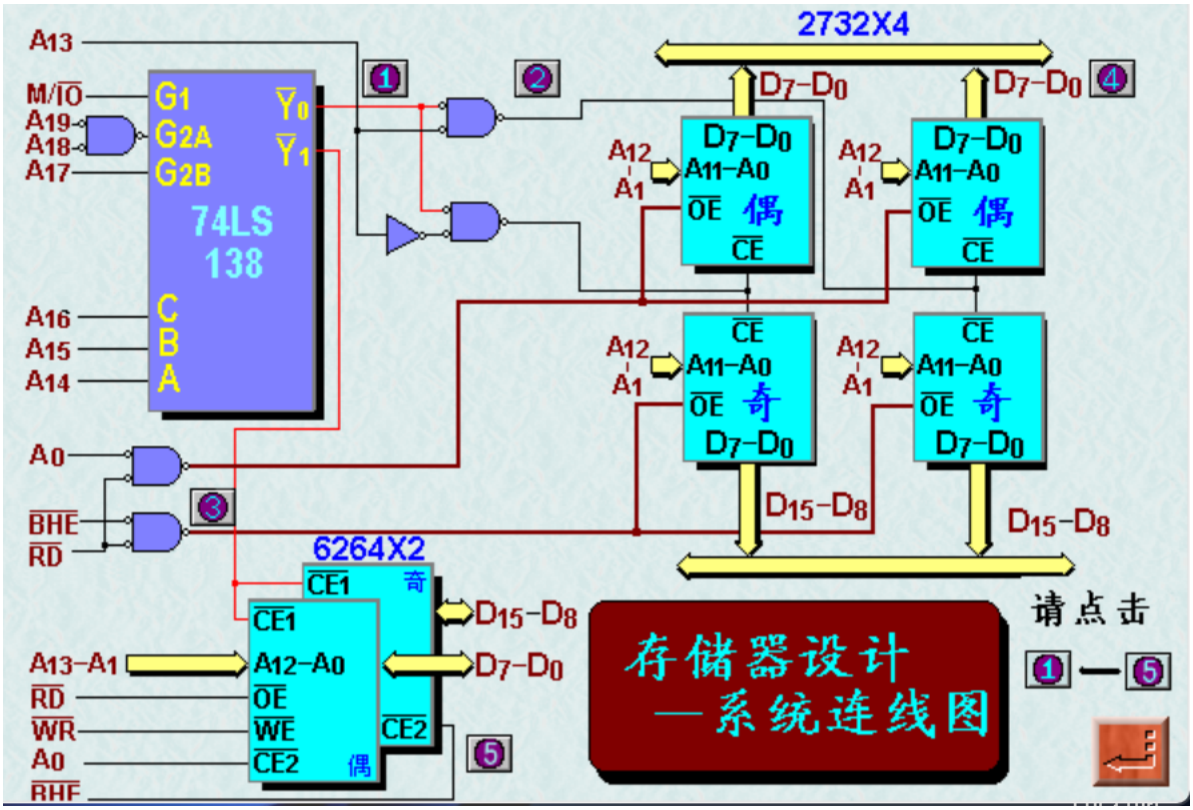

图 1.13: 存储器系统连线图

第二章　指令系统

2.1　寻址方式

2.1.1　立即寻址

操作数包含在指令中，它是一个 8 或 16 位常数，称为立即数。
示例

```
1    MOV AX,1860H
```

表示将 1860H 送入 AX 寄存器中。其中，立即数在代码段中，紧跟在操作码之后。

2.1.2　寄存器寻址

操作数包含在寄存器中，由指令指定寄存器名

```
1    MOV AX,BX
2    MOV CH,AL
```

表示将寄存器 BX 中的数送入 AX 寄存器中。第二条指令说明低位和高位可以互相传数字

2.1.3　存储器寻址

操作数在存储器中。这种方式速度慢，表达方式最多，情况最复杂，要求能在 1MB 存储空间中寻址。段内偏移量也称有效地址 EA。

物理地址计算方法

图示如下!!!（存储器地址计算图示）段寄存器可以是 CS,SS,DS,ES; 有效地址 EA 可以是

1. 指令中直接指定的 16 位直接位移量，要加方括号
2. BX/BP/SI/DI，出现 BP（不管是源处还是目的处）时默认使用 SS 提供段基址，但允许使用段超越前缀将 SS 修改为 CS/DS/E，没有 BP 的，默认使用 DS
3. 位移量 + 基址/变址

BX/BP +[SI/DI]+ 位移量，画斜线的两者不能同时出现

EA 的 24 种表示法!!! 图

2.1.4　其他

隐含寻址

指令中不明确指明操作数，但有隐含规定的寻址方式。如：DAA 对 AL 中的数进行十进制调整；CLI 对中断标志清零

一条指令包含几种寻址方式

例如

```
MUL DL
## AX<-AL*DL,源操作数为寄存器和隐含的寄存器AL
```

IO 端口寻址

直接端口寻址：端口地址由指令直接给出，范围是 00-FFH，若超出要先放到寄存器里。如

```
IN AL,40H
OUT 83H,AL
```

间接端口寻址：端口地址由 DX 寄存器提供，其范围为 0000-FFFFH

```
MOV DX,2F0H
IN AL,DX ##2F0H读入到AL
MOV DX,300H
OUT DX,AL ##AL输出到300H
```

转移类指令寻址

详见控制转移指令

2.2　数据传送

2.2.1　通用传送

MOV

将操作数（字或字节）传送到目标操作数，源可为 8 位/16 位通用寄存器（包括 AX-DX，SI，DI，BP，SP），段寄存器（CS 不能作为目的操作数），存储器或立即数。目的操作数不能为立即数，其他同源操作数。四种数据来源的传送关系如图!!! 下面是一些示例

```
# 立即数送存储器
MOV [BX],0FFH
# 立即数送通用存储器
MOV AL,30H
```

```
    MOV BL,'$'
    MOV AX,1250H
    MOV BX,OFFSET TABLE
    # 存储器和通用寄存器相互传送
    MOV [BP],BX # 带方框表示是段加偏移对应的存储器单元而不是BP本身的值受到改变
    MOV CL,5[BX]
    # 存储器与段寄存器之间相互传送
    MOV [SI],DS
    # 段寄存器和通用寄存器相互传送
    MOV ES,AX
    # 通用寄存器之间相互传送
    MOV AX,DX
```

常见错误

```
    # 立即数不能作目的地
    MOV 60H,AL
    # 存储器之间不能直接传送
    MOV [BX],[SI]
    # CS不能作目的操作数
    MOV CS,AX
    # 通用寄存器无IP寄存器
    MOV BX,IP
    # 段寄存器之间不能直接传送
    MOV DS,ES
    # 数的长度不一致
    MOV CX,AL
```

PUSH

将源操作数入栈并使 SP 减二.低位先入栈

```
    PUSH AX # 把AX入栈
```

POP

将当前 SP 指向的栈顶的一个字送到目的操作数中，并使 SP+2. 操作数可以是 16 位通用寄存器，DS/SS/ES 或存储单元。堆栈操作总是以字为单位

```
    POP AX # 先出栈的一个字节给到低位
```

XCHG

把字或字节的源操作数和目的操作数交换。可以发生在寄存之间或寄存器与存储器之间，但是段寄存器不能作操作数。

```
# Right
XCHG AX,BX
XCHG [BI],CL
# Wrong
XCHG AX,0AF4H #立即数不能交换
XCHG SS,[SI] # 段寄存器不参与交换
XCHG [SI],[BX+1] # 存储器之间不能直接交换
```

XLAT

将一个字节从一种代码转换成另一种代码。注意：

1. 使用指令前必须先创建一个表格，将转换表的起始地址装入 BX 中
2. AL 存表头地址到所查找的某一项之间的位移量，根据位移量从表中查到转换后的代码值送入 AL
3. 表中最多存 256 字节

典型的例子是用该指令将十进制数转换成七段显示管的代码

```
TABLE: DB 40H,79H,24H,30H,19H
       DB 12H,02H,78H,00H,18H # 做表
       ...
   LEA BX,TABLE # 把表头地址给BX
   MOV AL,5 # 把表内的偏移量给AL
   XLAT # 此时AL中有‘5’的七段代码12H(表内从零开始数)
```

2.2.2　输入输出

IN(OUT 同理)

把指定端口的内容读到 AL(for byte) 或 AX(for word) 中

```
# 格式1: 立即数指示端口
IN AL,nn #nn指代的8位端口的内容读到AL, nn=00-FFH
IN AX,mm #mm指代的16位端口读到AL, mm+1指代的16位端口读到AH.mm=00-FEH
# 格式2: 用寄存器给端口号
IN AL,DX # DX=0000-FFFFH
IN AX,DX # DX=0000-FFFEH DX指代的16位端口读到AL, DX+1指代的16位端口读到AH
```

当口地址大于 FFH 时，必须用格式 2。典型示例：扬声器发声。

2.2.3　地址目标

这是一类专门用来传送地址码的指令

LEA

取源操作数地址的偏移量，传送到目的操作数所在的单元 (load effective address)。源操作数必须是存储单元，目的操作数必须是除段寄存器以外的 16 位寄存器

```
# 注意区分LEA(取偏移地址)和MOV(取内容)
# 设SI=1000，DS=2000，(21000H)=1234,则
LEA BX,[SI] # BX=1000
MOV BX,[SI] # BX=1234
# 下面两段代码等效
LEA BX,TABLE
MOV BX,OFFSET TABLE
```

LDS

load DS: 从源操作数指定的存储单元中取出 4 个字节的地址指针送进一对目的寄存器中（常用 SI 寄存器，但是不能是段寄存器）load ES: 与 LDS 基本相同但是后两个字节给到 ES 而非 DS，目的操作数也常为 DI 而非 SI

```
# 设DS=0100，BX=0020，（01020H）=26A0H，（01022H）=B500H
LDS SI,[BX] # 低位两个字节给SI，SI=26A0，高位两字节给DS，DS=B500
LES DI,[BX] # DI=26A0，ES=B500
```

2.2.4　标志传送

LAHF

load AH flags: 将 SF,ZF,AF,PF,CF 送到 AH 的 7，6，4，2，0 位，而 5，3，1 位为任意值。

SAHF

send AH flags: 将 AH 的 7，6，4，2，0 位送到 SF,ZF,AF,PF,CF，而高位 flag（OF,DF,IF,TF）不受影响。

PUSHF/POPF

将整个标志寄存器的内容压栈/出栈，并使 SP-2/+2（仍然是低位先入栈，高位先出栈）

2.3　算数运算

总结图!!!

2.3.1　加法

ADD

目的数被赋值为“目的数 + 源”

```
# RIGHT
ADD AL,45H
ADD BL,DL
ADD [BX],CL
# Wrong
ADD 85H,AL # 立即数不能被赋值
ADD 5[BX],[BP] # 两个存储器内容不能直接相加
ADD BX,CL # 字节和字不能相加
```

加法指令结束后将影响六个标志位，如何解释这些标志位（是无符号数相加还是有符号数相加决定是进位还是溢出）取决于程序员。

ADC

add carry: 带进位的加法指令。目的 = 目的 + 源 +CF

INC

increment: 增量指令，对目的操作数加 1. 操作后影响 AF,OF,PF,SF,ZF，但不影响 CF

```
INC BL
INC BYTE PTR[BX] # 对内存字节单元内容+1
INC WORD PTR[BX] # 对内存字单元内容+1
```

AAA

ascii-adjusted add: 加法的 ASCII 调整指令。用 ADD 或 ADC 对两个非压缩十进制数或 ASCII 表示的十进制数作加法，且运算结果保留在 AL 中后，用 AAA 可以将运算结果调整为一位非压缩十进制数，仍然放在 AL 中。若 AF=1，则表示高位有进位，进到 AH 中。

* 非压缩十进数（BCD 数）：高四位全为零例题图!!!

DAA

decimal-adjusted add: 加法的十进制调整指令。将两个压缩的 BCD 数相加的结果调整为正确的压缩 BCD 数，相加结果必须在 AL 中才能使用 DAA。例题图!!!

2.3.2　减法

2.3.3　乘法

2.3.4　除法

2.4　逻辑移位

2.4.1　逻辑运算

2.4.2　算数移位

2.4.3　循环移位

2.5　字符串

2.5.1　传送

2.5.2　比较

2.5.3　扫描

2.5.4　装入

2.5.5　存储

2.6　控制转移

2.6.1　无条件转移

2.6.2　过程调用

2.6.3　条件转移

2.6.4　条件循环控制

2.6.5　中断指令

2.7　控制指令

2.7.1　标志操作指令

2.7.2　外部同步指令

2.7.3　停机和空操作